July 1980
$2.95

kilobaud

MICROCOMPUTING T.M.

Put PET on the BUS.
Mastering the IEEE 488.
P. 22

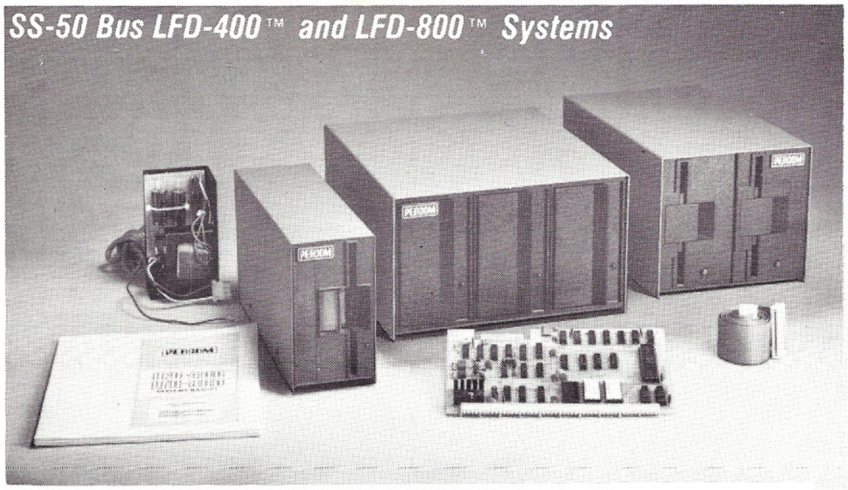
Your H8 Goes
2400 Baud and 64K.
PP. 81 and 90

Questions
and Answers
About Memory.
P. 164

A/D for the '80.
Easy-to-Build
Converter.
P. 206

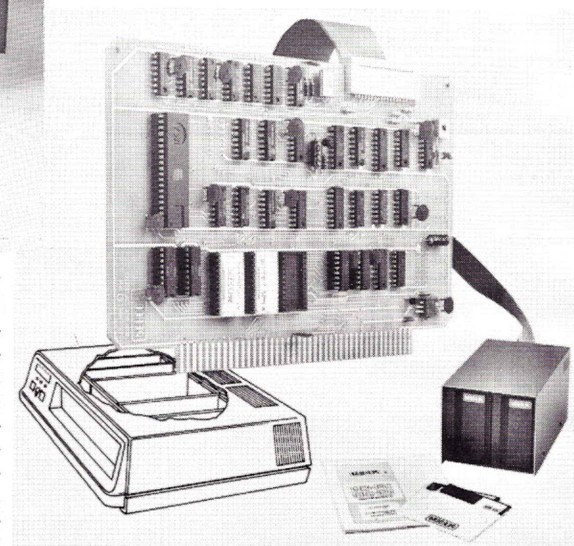
Commodore Goodies Reviewed.
New Computer, Floppies, Software. P. 36

A Few Extraordinary Products for Your 6800/6809 Computer



From Percom . . .

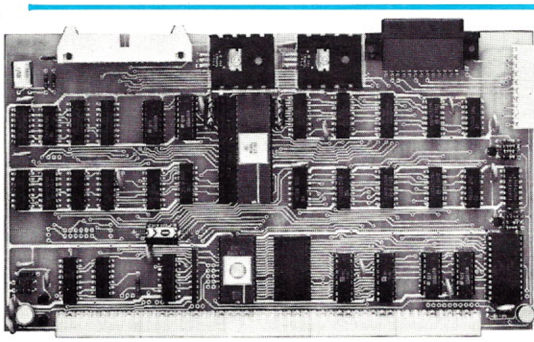
Low Cost Mini-Disk Storage in the Size You Want 14



EXORciser Bus LFD-400EX™ -800EX™ Systems*

Percom mini-disk systems start as low as $599.95, ready to plug in and run. You can't get better quality or a broader selection of disk software from any other microcomputer disk system manufacturer — at any price!

Features: 1-, 2- and 3-drive systems in 40- and 77-track versions store 102K- to 591K-bytes of random access data on-line • controllers include explicit clock/data separation circuit, motor inactivity time-out circuit, buffered control lines and other mature design concepts • ROM DOS included with SS-50 bus version — optional DOSs for EXORciser* bus • extra PROM sockets on-board • EXORciser* bus version has 1K-byte RAM • supported by extended disk operating systems; assemblers and other program development/debugging aids; BASIC, FORTRAN, Pascal and SPL/M languages; and, business application programs.



The SBC/9™. A "10" By Any Measure. 13

The Percom SBC/9™ is an SS-50 bus compatible, stand-alone Single-Board Computer. Configured for the 6809 microprocessor, the SBC/9™ also accommodates a 6802 without any modification. You can have state-of-the-art capability of the '09. Or put to work the enormous selection of 6800-coded programs that run on the '02.

The SBC/9™ includes PSYMON™, an easily extended 1-Kbyte ROM OS. Other features include:

- Total compatibility with the SS-50 bus. Requires no changes to the motherboard, memory or I/O.
- Serial port includes bit-rate generator. RS-232-C compatible with optional subminiature 'D' connector installed. 10-pin Molex connector provided.
- Eight-bit, non-latched, bidirectional parallel port is multi-address extension of system bus. Spans a 30-address field; accommodates an exceptional variety of peripheral devices. Connector is optional.
- Includes 1-Kbyte of static RAM.
- Costs only $199.95 with PSYMON™ and comprehensive users manual that includes source listing of PSYMON™.

Versatile Mother Board, Full-Feature Prototyping Boards 15

Printed wiring is easily soldered tin-lead plating. Substrates are glass-epoxy. Prototyping cards provide for power regulators and distributed capacitor bypassing, accommodate 14-, 16-, 24- and 40-pin DIP sockets. Prototyping boards include bus connectors, other connectors and sockets are optional.

MOTHERBOARD — accommodates five SS-50 bus cards, and may itself be plugged into an SS-50 bus. Features wide-trace conductors. Price: $21.95

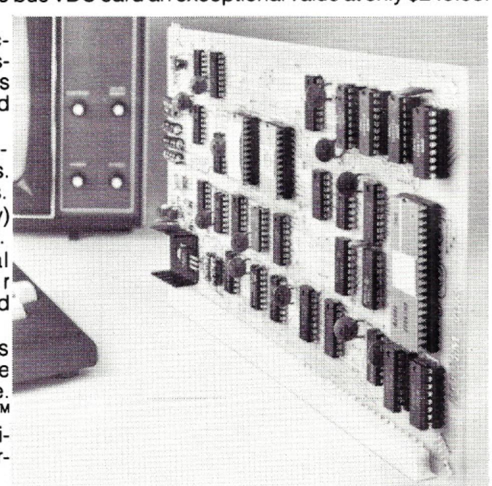
SS-50 BUS CARD — accommodates 34- and 50-pin ribbon connectors on top edge, 10-pin Molex connector on side edge. Price: $24.95.

SS-30 BUS CARD — 1¼-inch higher than SWTP I/O card, accommodates 34-pin ribbon connector and 12-pin Molex connector on top edge. Price: $14.95.

The Electric Window™: Instant, Real-Time Video Display Control 16

Memory residency and outstanding software control of display format and characters make this SS-50 bus VDC card an exceptional value at only $249.95. Other features:

- Generates 128 characters including all ASCII displayable characters plus selected Greek letters and other special symbols.
- Well-formed, easy-to-read 7x12-dot characters. True baseline descenders.
- Character-store (display) memory included on card.
- Provision for optional character generator EPROM for user defined symbols.
- Comprehensive users manual includes source listing of Driver software. Driver — called WINDEX™ — is also available on mini-diskette through the Percom Users Group.



™ trademark of Percom Data Company, Inc.
* trademark of the Motorola Corporation.
Prices and specifications subject to change without notice.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

Products are available at Percom dealers nationwide. Call toll-free, 1-800-527-1592, for the address of your nearest dealer, or to order direct.

INTRODUCING THE EMULATOR™

The trouble with video terminals today is that most of the low-cost models just don't have the performance to handle your tough applications. And the few that do are usually not compatible with your existing system. But now, Intertec has resolved this age old dilemma with the introduction of its new Emulator™ Video Terminal.

The $895* Emulator™ performs exactly as you command. With the depression of just a few keys, Emulator users can select terminal control codes of any one of four popular video terminals. The Lear-Siegler ADM-3A. The Soroc 1Q-120. The DEC VT-52. Or the Hazeltine 1500. Incredible! It's like having four terminals for the price of one.

But, best of all, not only does the Emulator replace these terminals, it outperforms them by offering enhanced user-oriented features. Features that those other terminals just don't have - at any price.

Standard Emulator™ features include: a sharp, crisp 12'' non-glare screen with a full 24 line by 80 column display. Twin RS232C serial ports - one for the host computer and one for your printer. Four separate cursor control keys. A separate 18 key numeric pad. Keyboard selectable baud rates and operating modes. And, a host of visual attributes.

No matter which dumb or smart terminal you're using today, don't buy another until you check out our new Emulator™. You'll get the performance of four terminals for the price of one. And you'll probably save hundreds of dollars over the price you paid for your last terminal. Plus, you'll get unparalleled reliability, nationwide service and quick delivery. Call or write us today for all the details. Intertec terminals are distributed worldwide and may be available in your area now.

INTERTEC DATA SYSTEMS®

2300 Broad River Rd, Columbia, SC 29210
(803) 798-9100 TWX: 810-666-2115

343

*Quantity one - Dealer inquiries invited.

QUALITY

THAT'S WHAT SEPARATES THIS PRINTER FROM THE TOYS

MICROTEK MT-80

SOLID VALUE FOR YOUR DOLLAR

The market is flooded with low-cost printers that look and last more like toys.

The Microtek MT-80, our versatile alpha-numeric line printer, has a high quality print mechanism that gives you solid value for your dollar. It has been designed with a superior brain resulting in more advanced features and more dependable performance. Our printer is so reliable that we offer you an incredible 365 days warranty.

We stand behind every printer we make because we build quality into each one. So stop tinkering with toys and get serious. Demand 100% value by specifying the MT-80.

OUR UNIT PRICE
$795 Parallel
$895 Serial (RS-232C)

LOADED WITH INNOVATIONS

- 40, 80 or 120 columns (software selectable)
- Non-thermal paper, pin feed
- 125 CPS, 70 lines per minute
- 9 × 7 dot matrix
- Vertical format unit
- 96-character ASCII (upper and lower case)
- Adjustable forms width to 9½″
- Parallel and serial (RS-232C) interfaces available

MICROTEK inc.

347

For more information contact:
MICROTEK, Inc.,
9514 Chesapeake Drive,
San Diego, CA 92123
Tel.(714)278-0633
TWX 910-335-1269

kilobaud

MICROCOMPUTING T.M.

contents: july '80

ARTICLES

DEPARTMENTS

Cover: Photo by Tedd Cluff. Bus compliments of Cheshire Transportation Co., Keene, NH.

micro info

$ This symbol next to a title in the table of contents indicates that the article is a business-application article.

Manuscripts

Contributions in the form of manuscripts with drawings and/or photographs are welcome and will be considered for possible publication. We can assume no responsibility for loss or damage to any material. Please enclose a self-addressed, stamped envelope with each submission. Payment for the use of any unsolicited material will be made upon acceptance. All contributions should be directed to the *Microcomputing* editorial offices. "How to Write for Microcomputing" guidelines are available upon request.

Editorial Offices:

Pine Street
Peterborough NH 03458
Phone: 603-924-3873

Advertising Offices:

Elm Street
Peterborough NH 03458
Phone: 603-924-7138

Circulation Offices:

Elm Street
Peterborough NH 03458
Phone: 603-924-7296

To subscribe, renew or change an address:

Write to *Microcomputing*, Subscription Department, PO Box 997, Farmingdale NY 11737. For renewals and changes of address, include the address label from your most recent issue of *Microcomputing*. For gift subscriptions, include your name and address as well as those of gift recipients. Postmaster: Send form #3579 to *Microcomputing*, Subscription Services, PO Box 997, Farmingdale NY 11737.

Subscription problem or question:

Write to *Microcomputing*, Subscription Department, PO Box 997, Farmingdale NY 11737. Please include an address label.

Kilobaud *Microcomputing* (ISSN 0192-4575) is published monthly by 1001001, Inc., 80 Pine St., Peterborough NH 03458. Subscription rates in U.S. are $25 for one year and $53 for three years. In Canada: $27 for one year only, U.S. funds. Foreign subscriptions (surface mail) — $35 for one year only, U.S. funds. Foreign air mail subscriptions — $62 for one year only, U.S. funds. In Europe, contact: Monika Nedela, Markstr. 3, D-7778 Markdorf, W. Germany. South African Distributor: KB Microcomputing, PO Box 782815, Sandton, South Africa 2146. Australian Distributor: Electronic Concepts, Attention: Rudi Hoess, 55 Clarence Street, Sidney 2000, Australia. Second-class postage paid at Peterborough NH 03458 and at additional mailing offices. Phone: 603-924-3873.

Wayne Green

PUBLISHER'S REMARKS

Watch Out—Here Comes Atari!

The lack of extensive advertising in the microcomputer media by Atari has tended to make this company invisible to many computer stores and even to many computerists. The system appears game-oriented, and comes from a game company. Even the name of the product, Atari Personal Computer Systems, works against it. "Personal computer" gives many people the impression that it must be something like a personal calculator. Obviously something "personal" has little to do with such serious subjects as business or education.

Yet, increasingly I see the Atari systems in department stores and electronics stores. One electronics store owner said he found Atari the most cooperative of the major microcomputer firms; he was quickly set up with a demonstrator and some stock. This happened while other manufacturers were still horsing around with credit applications and minimum order requirements.

The April 28 issue of *Newsweek* must have come as an eye-opener to many in the microcomputer business. There, was a four-page, front-cover, color foldout ad for the Atari systems! This sort of advertising support should help Atari develop its dealer network, enabling Atari to compete with Tandy's 7500 stores . . . a formidable marketing system.

On my recent trip to Europe I found it was more the exception than the rule not to find a microcomputer section in the major department stores. The micros are right in there with the typewriters and calculators—and selling well. Department stores don't invest in merchandise that just sits around. It *has* to sell or they can't afford the space and the inventory costs. In the case of computers, it also has to sell with a minimum of salesman time since there is no practical way for the larger stores to get trained computer help.

Atari's cartridge form of program transfer may be expensive for consumers, but it is simple to use and makes the Atari virtually self-demonstrating in stores. How much of a dent Atari will make in the microcomputer market remains to be seen.

The Texas Instruments Approach

While many computerists were disappointed with the TI-99 and felt that it probably would be withdrawn from competition, TI is pushing it, and the 99 is appearing in more consumer outlets. What did I see in Harrod's, the largest and most prestigious department store in England? The TI-99!

In the "Atari" issue of *Newsweek*, TI ran a two-page ad (pp. 30–31) for the 99 system. It's called a home computer, another term I feel is self-limiting for sales. I wish firms would remember that they are selling computers—small computers, to be sure, but they *are* computers. I prefer the term "microcomputers" since it doesn't limit the uses of the equipment in the imaginations of the prospective customers.

The TI ad, like Atari's, says virtually nothing about business use of the system. TI stresses the use of it at home for entertainment, education and finance. It compounds the "toy" concept by stressing the speech synthesizer, which allows the system to talk. TI also mentions that Milton Bradley—a name synonymous with toys—supplies the programs. The four-color "grabber" page picture, showing a group of people all beaming at you from behind a TI-99, and each holding some prop to indicate his interest, is similar to the old Imsai ads.

Other Ad Approaches

The best consumer ads I've seen have been those by Apple. They are attention-getting, and they must be prompting sales. I see the same ads running repeatedly in many of the better consumer magazines.

When I look at the Instant Software sales records I see that the top sellers are utilities and educational/game programs. It's something to think about when you put together ads for the general public.

Sherry Smythe

OUTPUT FROM ISI

Report from Europe

The growth of the European market for microcomputer software makes clear the need for a plant to produce Instant Software in Europe. Talks with firms in the field hinted that our best bet would probably be in England or Ireland. Recent construction of plants for Centronics and Memorex in Ireland suggested we check there first.

Ireland has benefits that other countries don't. The Irish want new industry enough to foot part of the bill for setting up a new plant. They will pay the total cost for training their people to work in these plants. In addition to those incentives, they offer a ten-year holiday on taxes—plus another ten years of very low taxes on profits after that. Wayne and I went to Ireland to visit the Industrial Development Authority (IDA) and look at some plant sites, one in Dundalk, near Northern Island.

From Ireland we went briefly to London, and then on to Germany and the incredible Hannover Fair. This huge fair attracts over 500,000, and most of the microcomputer firms, including Instant Software, exhibit there. We got a first look at the new IBM personal computer.

We also visited computer stores and Tandy stores. In many, people had the idea that American business programs would not translate well for European use because the business systems were different. We looked into that possible problem and found that only some of the accounting systems were different. Most businesses encounter the same programming needs, no matter what the country. The big need in Europe is for business-applications programs.

While visiting a Tandy store in Rotterdam, Holland, we located someone to do the Dutch translations of Instant Software programs. In Amstelveen, Holland, we viewed the Hong Kong-built TRZ-80 system, which operates on TRS-80 software and comes complete with Microsoft BASIC.

When we went to Belgium, we found a small-town Tandy store selling three TRS-80 systems a month. Microcomputers are gaining steam in Europe. We were surprised at the penetration of TI; we saw TI systems on sale in London, Germany and France.

Tandy is limited by the number of their stores, but they are developing Tandy Computer Stores, with, I think, four in place so far. We stopped at one in Köln that was selling about 25 systems a month. Not bad!

Robert W. Baker

PET-POURRI

A Multiprocessor Printer

XYMEC, a subsidiary of Litronic Industries, recently announced a revolutionary "intelligent" printer, for personal and small-business computer applications, that will interface to the PET.

The HY-Q 1000 is a low-cost, letter-quality daisy-wheel printer with five internal microprocessors. Simple "control" codes provide many commonly used text formatting functions and eliminate the need for complex, memory-consuming software packages. The average throughput print speed is 30 characters per second, and the printer has a full function keyboard.

The HY-Q 1000 provides "Quadra-Pitch" with 10, 12 or 15 characters per inch and up to 198 characters per line. It also has proportional spacing and prints 100 printable characters in five languages without changing the daisy wheel. Another unique feature is its reverse printing, with white characters on a black background. You can even choose from 21 type styles and five ribbon colors.

Other interesting features include underlining and bold printing, right justification, decimal point location, title centering, auto column layout mode, auto printing of vertical lines to separate columns, auto (paragraph) indent mode, full electronic tab functions and auto storage/recall of often-used phrases and formats.

The printer retails for $2495 and is based on an Olivetti electronic typewriter. It comes with a three-month warranty, and maintenance is readily available at over 500 Olivetti service centers throughout the United States.

For more information, contact XYMEC, 17791 Skypark Circle, Suite H, Irvine, CA 92714.

Quick Reference Card

The price of the PET Quick Reference Card was listed incorrectly in the December column. The correct price is $3.50, postpaid, from Leading Edge Computer Products, 4471 Santa Monica Blvd., Los Angeles, CA 90029.

The reference card has nine pages of condensed notes and information on the PET folded to a handy pocket size. It covers the operators, special variables, monitor commands, BASIC commands and statements, ASCII character values, I/O device addresses and status bits, functions, abbreviations and useful memory locations for all models. A second printing is planned, and should have more information on the PET printer and disk.

I hope Leading Edge considers adding the BASIC token values that are not in the current version, since I use these quite a bit. Otherwise, the reference card is fairly complete and contains nearly everything you would ever want.

Spacemaker Update

The new PETs and CBMs provide three sockets on the primary logic board for ROM expansion. They are addressed as 4K blocks with a starting address of $B000 (hex). Several ROM-based products use the same address space and therefore cause a conflict if more than one is desired. Specifically, the Commodore Word Processing Package, the BASIC Programmer's Toolkit and the Computhink DOS all use the first available expansion socket.

In my last column I mentioned Small System Services. SSS's Spacemaker is a utility device for new PETs that is designed to allow user selection between ROMs that must occupy the same address space. The companion ROM-driver and user I/O packages allow simple software selection of various groups of ROMs installed in one or more Spacemakers.

I recently received a sample Spacemaker along with more complete documentation. The unit impressed me. The documentation and hardware are both excellent.

The Spacemaker board is only 4.25 × 2.25 inches and plugs vertically, by a special header, into any of the ROM expansion sockets inside the new PETs. A pair of conflicting address ROMs are then inserted in the two 24-pin IC sockets on the Spacemaker board, and the installation is complete.

The board also has a short cable attached to a miniature toggle switch that mounts conveniently on the outside of the PET. The switch is actually mounted using a pressure-sensitive mounting clip, so no drilling or modifications are required. This switch provides manual selection of either of the two ROMs mounted in a single Spacemaker. Since Spacemakers were designed to work alone or in tandem, additional units can be added to switch pairs or groups of ROMs if necessary.

For even greater flexibility, an extra connector on the Spacemaker board provides an interface for software control of ROM switching. This normally requires either the user I/O kit or the ROMdriver controller board. The user I/O kit includes a simple user port connector and a utility program provided on diskette. If your user port is already dedicated or used frequently, the ROMdriver controller board provides identical functions without using the user port.

kilobaud
MICROCOMPUTING T.M.

PUBLISHER/EDITOR
Wayne Green

ASSISTANT PUBLISHER/EDITOR
Jeffrey DeTray

MANAGING EDITOR
John Barry

EDITORIAL ASSISTANTS
Dennis Brisson
Susan Gross
Eric Maloney

ASSOCIATE EDITORS
Frank Derfler, Jr.
Rod Hallen
Peter Stark
Sherm Wantz

PRODUCTION DEPARTMENT
MANAGER:
Noel Self

ASSISTANT MANAGER:
Robin Sloan

STAFF:
Steve Baldwin
Tedd Cluff
Robert Drew
Bruce Hedin
Kenneth Jackson
Ross Kenyon
Clare McCarthy
Michael Murphy
Dion Owens
Robert Sawyer
Patrice Scribner
Susan Symonds
John White

PHOTOGRAPHY
William Heydolph
Terrie Anderson
Reese Fowler

TYPESETTING
Barbara Latti
Sara Bedell
Mary Kinzel
Karen Podzycki

PROJECTS EDITOR
Jim Perry

BOOK EDITORS
Chris Brown
Emily Gibbs

EXECUTIVE VICE PRESIDENT
Sherry Smythe

CORPORATE CONTROLLER
Alan Thulander

EXECUTIVE ASSISTANT
Leatrice O'Neil

ACCOUNTING MANAGER
Knud Keller

CIRCULATION MANAGER
Debra Boudrieau

CIRCULATION
Barbara Block
Pauline Johnstone

BULK SALES MANAGER
Ginnie Boudrieau

AUSTRALIAN DISTRIBUTOR
Rudi Hoess

EUROPEAN MARKETING DIR.
Reinhard Nedela

ADVERTISING
603-924-7138
Kevin Rushalko, Mgr.
Marcia Stone
Penny Brooks
Jerry Merrifield
Hal Stephens

A single installed ROMdriver or user I/O package can serve as the resident controller for one, two or three Spacemakers, with ROM selection and switching via software. But whenever more than one Spacemaker is used, additional cables are required and are optionally available.

SPACECTL, the utility program provided, maintains a user ROM menu and displays a list of the available utility software as defined by the user. It will show which items use a particular ROM and whether that ROM is enabled, disabled or under manual control. When you select a utility, SPACECTL will enable the selected software and set ROMdriver to enable or disable the necessary ROMs. SPACECTL can be easily adapted for different configurations, more Spacemakers and new or different utilities by simply modifying the menu table itself.

A Computhink option is also available for the new Computhink disk drives, with DOS in masked ROM only. Since Computhink already uses the PET user port for printer output, both the Spacemaker and ROMdriver are suggested. But only one Spacemaker can be used.

For detailed pricing, availability and more information on options and configurations, contact your local PET dealer or write Small Systems Services, Inc., PO Box 5119, Greensboro, NC 27403.

Programming Ideas and Tips

Here's another interesting subroutine from the old 8K ROM operating system. This one will search the BASIC program variable table looking for a specific variable and return a pointer to the variable. By reserving certain variables for specific purposes, you can easily pass multiple parameters between machine-language routines or user functions and the main BASIC program. This can be much simpler than peeking and poking values into fixed memory locations, since you may have to convert values before using them. By using standard BASIC variables to pass parameters, the values are always in the correct form and ready to use.

To use this subroutine, you must first understand how the variables are stored in memory by BASIC. All simple variables or variable pointers are stored as 7-byte values in the variable table. Disregarding arrays, the first two bytes of each entry identify the two-character variable name and the variable type as either integer, floating point or string (see Table 1).

The remaining five bytes will either contain the value of the variable or a size and address pointer to the string (see Table 2).

For more information, see the section on variable allocation at the back of your PET user manual. If you have a new ROM set or a 16/32K PET, then the format shown in the manual for arrays is incorrect, since the limit of 255 array elements was corrected.

Now back to using the subroutine to find a given BASIC variable. Before the subroutine can be called, you must set the variable you are searching for in locations $94 and $95 (148 and 149 decimal). This corresponds to the first two bytes of each 7-byte entry in the variable table (Table 2) with the flag bits to indicate the variable type.

You then call the subroutine at location $CFDB (53211 decimal). On return, locations $96 and $97 (150 and 151 decimal) will contain the address of the first byte following the 2-byte variable name in the variable table. Thus, this address can be used to directly set the value of a floating-point or integer variable or to get the length and starting address of a string.

One word of caution on using this routine. The variable should be defined somewhere in the BASIC program prior to being searched for by this routine. If not found, the variable will be added to the variable table with a default zero or null string value.

For those who have the new ROMs or new PETs, I'm sorry I haven't had time to track this routine down in the new ROMs. If someone is able to locate it, please let me know.

Variable type	Byte-1	Byte-2
Integer	char1 + 128	char2 + 128
Floating Pt	char1	char2
String	char1	char2 + 128

Note: If the variable is a single character name, char2 will have a zero value.

Table 1.

Variable type	Byte-1	Byte-2	Byte-3	Byte-4	Byte-5
Integer	256*HI	LO	0	0	0
Floating Pt	[--- actual floating point value ---]				
String	#chars	ptr-HI	ptr-LO	0	0

Table 2.

Programming Style

Over the past year or so, I've helped Instant Software with their reviews of several programs written for the PET. One of the biggest problems I've seen with the software submitted is the lack of checking for valid responses to questions.

It usually doesn't take much added effort to have the program check all user input. It can be very frustrating to a user if the program stops because he typed in a bad value or accidentally hit the wrong key.

You should always use a string variable for user response and convert strings to numbers when required, using the VAL function. This allows you to check for valid responses in all cases before trying to use a value. It also prevents the confusing REDO FROM START error message if you enter a letter when the program expects a number.

If only a single character response is required, use the GET function to eliminate the return required by an input:

```
100 GET R$ : IF R$ = "" THEN 100
```

This also eliminates the problem of stopping the program if the return key is hit without entering any data in response to an input. This latter problem can also be avoided by printing a default response before asking for input. The best method I've seen involves printing the normal question with three additional spaces and the half-box character (■) generated by the shift quote key. Then print three cursor lefts to replace the cursor to the normal position after the question and ask for input. The cursor will appear as a half box alternating from top to bottom and back. If the return key is not hit without entering any other data, you'll receive the half-box character as a default input and can substitute another value if desired:

```
100 PRINT"FILE NAME [3-spaces] ■ [3-cl]";
110 INPUT N$
120 IF N$ = "■" THEN N$ = "DEFAULT"
```

You could also use this method to set up a real default answer such as Y or N, then simply hit the return key to use the preset default value:

```
100 INPUT"PRINTED COPY [3-spaces] N [3-cl]";A$
```

When using the GET command, a simple subroutine will accept any one of a predetermined set of valid responses. By setting a variable to contain a string of valid characters before calling the common subroutine, the subroutine will then accept only valid characters before returning:

```
100 R$="NSEWUD" : GOSUB 5000
.
.
240 R$="YN" : GOSUB 5000
.
.
590 R$="12345" : GOSUB 5000
.
.
.
5000 GET C$ : IF C$ = "" THEN 5000
5010 FOR X = 1 TO LEN(R$)
5020 IF C$ = MID$(R$,X,1) THEN RETURN
5030 NEXT : GOTO 5000
```

This subroutine will only accept characters defined in R$ prior to returning.

One other problem I saw was in a menu-driven program. The program displayed a list of options and asked the user to enter the number of the option desired. In this particular program, one option had to be used prior to selecting certain other options to set up specific values.

The problem was that there were no checks that the proper sequence had been followed; the program could generate erroneous results when the improper sequence was used. A simple solution would have been to add a variable as a flag that set to a nonzero value when the required sequence was executed. The flag could then be tested before trying to do a particular operation that depended on prior data being established.

When you write a program for others to use, you should be extremely careful with the user interface. Remember, you wrote the program and should know what it's doing. The user is completely ignorant as to what is being done inside the program and cannot be expected to know what you are expecting him to do.

Try to make the program as idiot-proof as possible, even if it seems silly. Don't let the user do anything you don't want him to do. Then, before submitting a program to any organization for review, have someone who knows nothing about the program try it.

Recent Letters

In the January issue, I gave information on how to use a Block Move routine in the 8K BASIC ROM. At that time I did not have the new ROMs and did not have the corresponding information. Mike Munchy of Brooklyn, NY, found the routine in the new ROMs and sent the information in Table 3.

For more information on this routine, see January's PET-pourri.

Several people commented that they didn't see a need for the BASIC line number search routine. Think about this one: how about storing small amounts of data within DATA statements to eliminate tape data files? . . . how about deleting game instructions printed by a subroutine at the end of a program and replacing the entire subroutine with a simple RETURN command? You'll need to know where the BASIC lines are located in memory, won't you?

OLD ROM . . .	NEW ROM . . .	FUNCTION . . . .
\$A9-\$AA	\$57-\$58	upper limit—old area to be moved
\$AE-\$AF	\$5C-\$5D	lower limit—old area to be moved
\$A7-\$A8	\$55-\$56	upper limit—new area to move into
\$C2E1	\$C2DF	routine starting address

Note: all values are shown in hexadecimal.

Table 3.

Anyway, think about it. I'm confident someone out there will discover a good use for the routine. If you come across any new ROM routines you think may be useful, drop me a line and we'll spread the good word.

Another letter came in from Preston Marshall of Walpole, MA, explaining the Axiom printer direct speaker control along with a sample program. A CHR\$(6) turns the speaker on by energizing the speaker electromagnet. A CHR\$(5) will then turn the speaker off. After locking the printer on the IEEE bus with a CMD instruction, putting a delay between the two speaker control characters and repeating the action for a period of time will produce various sounds.

Because of the execution time of BASIC, you cannot get any high-pitched tones from the printer speaker. It can't compete with the quality of sound produced by the normal PET sound interface (the user port CB2 line), but it might be useful for some purpose.

Anyone with an Axiom printer who's interested can try the following program to see what the printer speaker will do:

```
100 INPUT "ENTER TIME DELAY (1-300)"; M
110 OPEN 4,4 : CMD 4
120 FOR I = 1 TO 1000
130 PRINT CHR$(6);
140 FOR K = 1 TO TM : NEXT K
150 PRINT CHR$(5);
160 NEXT I
170 PRINT#4
180 CLOSE 4
```

Please do not send PET-pourri correspondence to *Microcomputing*. We have to turn around and forward it to the author, which wastes time and postage. If you want to correspond with Bob Baker, please write directly to him at 15 Windsor Drive, Atco, NJ 08004. Thank you.

BOOK REVIEWS

Learning Level II

David A. Lien
Compusoft Publishing, Inc.
San Diego, CA, 1979
Softcover, 352 pp., \$15.95

In 25 chapters and four appendices, author Lien and editor Dave Gunzel have combined talents once more to produce another important installment in their Everyman's series of microcomputer programming instruction books.

Chapter 5, "Chasing the Errors," and Appendix A, "Level II Error Message Reference Guide," are particularly valuable.

Chapter 5 takes much of the mystery out of error trapping by explaining how to use ON ERROR GO TO, ERL and ERR/2+1 statements.

In Appendix A, Lien explains the 23 error codes to which the TRS-80 Model I responds. More significantly, he provides a series of test programs that demonstrates how each error is generated and corrected.

The book also lists changes that can be made with pen and ink in the *User's Manual for Level I* to make that book—another Lien/Gunzel work—compatible with Level II BASIC.

A second section can be removed from the book, cut into parts and pasted into the Level I manual.

Chapter 1, "Level II Overview," got my attention immediately by explaining how hard-to-find syntax errors can be created unwittingly by use of a variable name that includes one or more of the 118 reserved words used in Level II BASIC and disk BASIC. Even using simple, two-letter variables such as ON, TO, IF and OR can cause syntax errors.

In the two chapters that explain how to use the Level II editor, I learned that the editor is not part of the BASIC interpreter. It is a special feature that is called using the word EDIT.

I learned also that while in the edit mode, I could type L after each insertion, deletion and so on (before pressing ENTER) to view the current condition of the line being edited. Somehow, I had missed that subtle point in the *Level II BASIC Reference Manual.*

Other chapters in Lien's book for nonprofessional users of Level II BASIC deal with searching and sorting routines, special uses of the INKEY\$ function, applying PRINT USING statements to control both numbers and strings and using PEEK and POKE.

Lien presents sample programs to demonstrate the use of commands, statements and functions. He introduces errors into each program to highlight the inherent limitations of the BASIC interpreter; however, he always follows the mistake with a correction.

Also included in the book are chapters that teach TRS-80 Model I owners how to convert programs from Level I BASIC to Level II BASIC, use the Radio Shack expansion interface, operate the real-time clock that comes with the interface unit and operate dual cassette recorders.

As he did in the *User's Manual for Level I,* Dr. Lien, an uncommonly gifted educator, employs a light, folksy style to disarm his readers and make them more receptive. He makes learning fun. For example, he introduces DEFSTR (to define specific variables as strings and eliminate the need to append the "\$" to the variable) as a statement to be used by thrill seekers only, claiming that DEFSTR will be loved by those who attract trouble.

In another sample program, he uses string variables containing DATA statements without first clearing memory space (beyond the 50 bytes that are automatically cleared). He then allows the reader to run the program until an out-of-string space error (?OS error) occurs. He proceeds to explain how the error occurred and why.

Learning Level II was an educational experience for me in many ways. Within a month

from the time I had received my copy of this book, I received an errata from the publisher. It contained corrections for typesetting and programming errors that had appeared in the printed book. Never before had I received such consideration from a publisher.

Every owner of a TRS-80 Model I system should own and use a copy. It will prove useful as both an instruction manual and a reference book. And—unlike the *Level II BASIC Reference Manual—Learning Level II* comes complete with an index.

Sherman P. Wantz
Sebring, FL

PIPS for VIPS, Vols. I and II

Tom Swan
Aresco
PO Box 1142, Columbia, MD
$19.95 each with tape cassette

VIP owners will find the first two volumes of *PIPS* invaluable sources of information.

I got Vol. I while wrestling to get a 4K program into 2K. *PIPS* included a CHIP-8 editor, just what I needed. Instead of a single byte, I can now see five instructions. The editor can also easily relocate blocks of data.

The same volume included a text editor, an 1802 machine-language disassembler, a character designer and two game programs.

The text editor is moderately handy with such functions as scrolling, cursor and insert. It allows six pages, each containing 16 lines of 16 characters.

The disassembler gives assembly-language mnemonics from the machine-language programs.

With the character designer, I can create my own display characters. I've been so busy writing, editing and disassembling programs, I haven't had time to try the games!

Aresco came out with Vol. II a few months later. This time Tom Swan had created something even better than the editor—a CHIP-8 assembler.

The assembler uses the text editor for entry of the source program with labels and comments. It outputs a symbol table, a link table and an object code on tape. These outputs include addresses calculated from the labels in the source code.

Vol. II also included the software modifications needed to use an ASCII keyboard with the text editor.

The loose-leaf books (without binders) are obviously not high-volume productions, but they are well-written, and often more thorough than RCA's VIP manual.

I do have a few criticisms. First, some of the alphanumeric characters used in the text editor are strange. Given the limited display resolution of the VIP, though, it's hard to do better.

Second, *PIPS*, Vol. II, assumes you just plug in an RCA keyboard.

Third, Aresco sent me the Vol. II tape without any information on the contents. I hope they've corrected this error in later mailings.

Although you can buy the books without the tapes for $14.95 each, don't skimp. Without the tapes, you'll spend days punching in the programs. Also, the symbols for display are referenced several times, but given only in Vol. I.

Joe Everhart
Brooklawn, NJ

TEA: An 8080/8085 Co-Resident Editor/Assembler

Dr. Christopher A. Titus
Howard W. Sams & Co., Inc.
Indianapolis, IN, 1979
Paperback, 254 pp., $8.95

If you've ever gone into a computer store and asked for a low-cost assembler, you probably couldn't help notice the salesmen in the back laughing at you.

Let's face it—low-cost assemblers are almost nonexistent. When you do find one, you discover that the $19.95 assembler requires a $129.95 text editor.

Fortunately, Dr. Titus has changed all that. *TEA: An 8080/8085 Co-Resident Editor/Assembler* describes a text editor and assembler program designed to work on 8080 and 8085 microcomputers. The text editor and assembler reside in memory simultaneously, therefore taking a fraction of a second to jump from one program to the other. Hence, creating source programs, assembling them and correcting them becomes simple.

The TEA (Tychon Editor-Assembler) text editor has an impressive number of features and commands. Titus thoroughly explains the commands and gives an example of each. They include APPEND, which allows text to be added to a source program; INSERT, which allows one or more lines of text to be inserted between two others; DELETE, which erases a line; and SEARCH, which will search the entire source program for a specific string.

Other commands save or load the program onto paper or cassette tape, change memory boundaries and call special user-written subroutines. The editor also features an auto-line numberer and several error messages to prevent you from accidentally destroying your program.

The TEA assembler, which converts the source program into machine code, passes over the source program three times.

The first pass generates a symbolic address table, which contains the 16-bit addresses of the various subroutines, jump destinations, buffer locations and so on.

During the second pass, the mnemonic op codes are converted to the binary values that the microprocessor can understand and execute.

Finally, the third pass is used to create a listing of the assembled program containing the op codes and data bytes in either octal or hexadecimal.

The assembler features a number of error messages and will specify on which line the error occurred. Pseudo op codes can be used, and labels, statements and comments don't need to be in any rigid format. Thus, a label can be anywhere in a line without an error occurring.

In addition, you can have several independent programs in the memory simultaneously. Should you have a source program that's too big for your memory, you can break it into several subprograms and merge them during the assembly process.

TEA's I/O subroutines will interface CRT terminals, audio cassettes and paper tape readers and punches. Should you find these routines incompatible with your I/O devices, you should be able to modify them. You can also easily add other I/O routines for such things as floppy disks. Once you've added the new subroutine, you simply change two bytes in TEA's jump table and you're done.

The book's appendices include a description of the TEA paper tape format, an explanation of how to modify the Z command, a description of how to add nine user commands and a summary of TEA's 17 commands.

The assembly listing (Appendix E) contains memory addresses, op codes and mnemonic codes and is lavishly commented. In fact, by carefully studying the comments, you should easily be able to modify TEA to suit your personal needs.

TEA is a comparatively large program; it resides in slightly over 6K of memory. The minimum recommended configuration is 8K.

As of this writing, no one offers TEA on paper tape, floppy disk or cassette tape. You will therefore have to hand-load TEA into your memory, a tedious and slow process. It took me over 20 hours to relocate and load the program, debug it and modify the I/O routines.

Since TEA is written in 8080 machine code, it will also work on 8085 and Z-80 systems (it will not be able to use Z-80 mnemonics). If you own several different systems, it should be easy to convert TEA into a cross-assembler that generates op codes and mnemonics for other CPUs.

I do have two complaints. First, any instructions in the source program that involve memory references (e.g., JMP, CALL, LXI SP) require three lines. The first line contains the op code, the second contains the operand and the third must contain a Ø. The reason for this is when the assembler prints a listing of the assembled program, it prints one byte per line. Hence, two-byte instructions require two lines, and three-byte instructions require three lines. The Ø reserves a line for the third byte. If the Ø is left out, an error will occur; this can be a problem for those who have worked on assemblers that use a different format.

Second, I was hoping the book would discuss how assemblers work so I could develop one for my 1802 system. But it is closer to a user's manual than a how-to book.

Nevertheless, if you are interested in how text editors or assemblers are written, I suggest you purchase this book anyhow, since it does have one of the few complete listings of such programs that I have seen anywhere. If you are studious, you should be able to figure out how TEA works by carefully examining the listing and its numerous comments.

Steve Dominguez
Golden, CO

Edited by Dennis Brisson

NEW SOFTWARE

Music Interpreter

Musicraft is a development system that turns your computer into an easy-to-use interactive musical instrument on which to create, store and perform music. It produces up to four voices, each with a seven-octave range, each with different musical sounds. It supports measures, musical line numbers, repeats, refrains, key signatures, accidentals or microtones. An advanced screen editor features simple notation and automatic error detection. The five integrated machine-language programs are: Setup, Editor, Waveform Generator, Interactive Play and a two-pass Compiler.

Musicraft requires an 8080, 8085 or Z-80 S-100 bus computer with a 24K CP/M system, a terminal with addressable cursor and the Newtech Model 6 Music Board. It is available on 8-inch single-density IBM, single-density North Star and Micropolis Mod II CP/M formats. Price is $79.95.

Newtech Computer Systems, Inc., 230 Clinton Street, Brooklyn, NY 11201. Reader Service number 483.

Record Selection for MWP Files

SELECT/MWP includes the Select program, which copies selected records from a mini word processing (MWP) name and address file to a new file. The selection criteria are interactively defined by the operator and allow complex selection tasks based on one or more of the variable values in each record. Any number of selection requests may be specified.

System requirements include an 8080 or Z-80 mainframe with 48K of memory, floppy or hard disk and CRT. It operates on CP/M with Microsoft MBASIC or Mits/Pertec Disk Extended BASIC. The MWP system is required for use of Select. Price is $95.

The Software Store, Ltd., 706 Chippewa Square, Marquette, MI 49855. Reader Service number 492.

Forms-Control Word Processor

DataStar word-processing software is a forms-control-oriented, general-purpose data entry, retrieval and update system specifically designed for office personnel usage. The keyboard input format is designed for high-volume production by clerk typists, and its dynamically displayed menu of operator prompting is designed for easy learning.

Complete control is provided by visual feedback over forms design, individual field length and recording attributes, data entry order, data verify procedure (selective field verify by field order, by sight, retype or list match), disk record format, as well as arithmetic processing using any combination of field, constant and file-derived numeric data. Completely CP/M compatible, DataStar operates with any CRT terminal or video board possessing addressable cursor and 32K–48K RAM. Data files constructed by DataStar are compatible with any CP/M-MP/M-supported language. Price is $350.

MicroPro International Corporation, 1299 Fourth Street, San Rafael, CA 94901. Reader Service number 478.

Imagination Business Program

The Personal Business Machine is a new business program for the Imagination Machine from APF Electronics, Inc., 444 Madison Ave., New York, NY 10022. It allows you to calculate financial status and compute loan amortization, interest and principal payments and interest rates. It computes exactly how much you can save by paying off the balance of your loan this month, or it will help you determine how much profit an investment will yield within a certain time period. Reader Service number 490.

North Star Menu Builder

MENUBldr, written in North Star BASIC, permits a user to generate screen menus in 2–3 minutes. Adding additional items or changing the format takes less than a minute. In a ten-menu application program, the user will typically save three to ten hours of development time.

Control information, displayed after every change, is user specified for each menu. The user thus sees the actual menu format without running the application software. Changing an existing menu requires only changing the desired menu control. The program stores the menu control values, menu title and menu text on a disk. Menus can be shared between programs, which is useful for standardizing formats. The companion program, MENUPRNT, is used with the user's application software display's previously created menus. The application software is appended to MENUPRNT. Menus may be displayed on a CRT or printer. MENUBldr requires a 24K RAM system with a disk drive. Price is $48 for source code on a single- or dual-density disk.

American Planning, 4600 Duke St., Suite 425, Alexandria, VA 22304. Reader Service number 481.

Real Estate Program

REILEY, a real estate index and locator, is a system of BASIC programs designed to help real estate agencies find properties for customers. These interactive search and sort programs automatically match the home specifications of a prospective buyer with current Multiple Listing Service (MLS) listings. No complicated codes or symbols are required to access information, nor is computer expertise necessary.

REILEY can give comparisons of mortgage terms, access detailed property descriptions, print complete amortization tables, estimate the monthly payment and sale price that the buyer can afford and determine the replacement price of any home on the MLS file. To operate REILEY, you need a single-drive, single-density system with 24K of user memory.

The Electric Abacus, 19 Mayfield Rd., Regina, Saskatchewan S4V 0B7, Canada. Reader Service number 499.

PET Word Processor

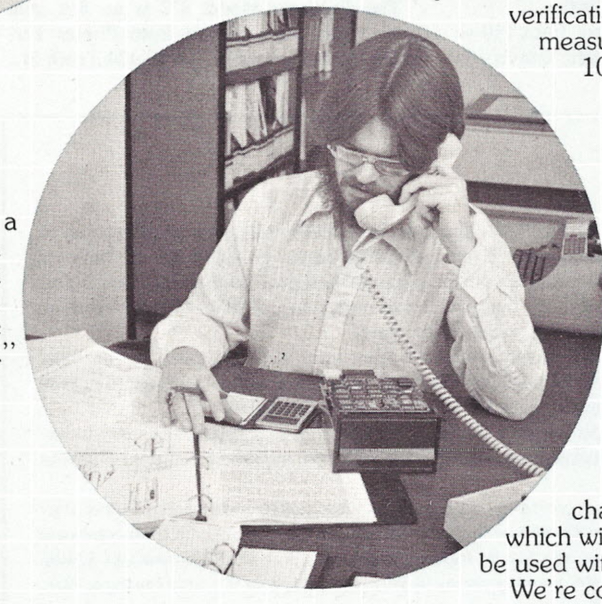
Paper-Mate Command 60, a full-featured word processor, incorporates full-screen editing with graphics for all 16K or 32K PETs, printers and disk or tape drives using 60 commands. It also includes most features of the CBM WordPro 3.

For writing text, Paper-Mate has a definable keyboard so you can use either business or graphics machines. Shift-lock on letters only, or use the keyboard shift lock. All keys repeat. Text editing includes floating cursor, scroll up or down, page forward or back and repeating insert and delete keys. Text block handling includes transfer, delete, append, save, load and insert. All formatting commands are embedded in text for complete control. Unlike most word processors, PET graphics as well as text can be used. Paper-Mate can send any ASCII code over any secondary address to any printer. Price is $29.

A B Computers, 115 E. Stump Road, Montgomeryville, PA 18936. Reader Service number 486.

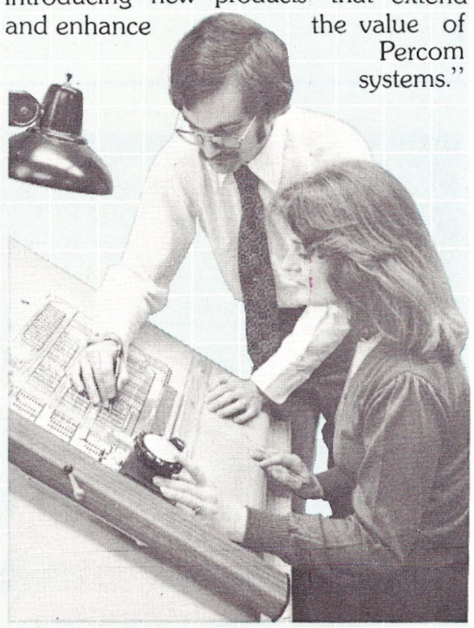
"From an efficient 1K-byte control system DOS to high level languages such as FORTRAN and Pascal, no other microcomputer disk systems manufacturer provides the range and quality of development and application programs available from Percom."

"Connie is running a 'cats eye' test on a mini-disk drive to check radial track alignment. Drive motor-speed timing and sensor alignment tests have already been performed. Disk formatting and format verification tests are next. These measurements are part of the 100% testing every single unit receives."



"Whether you call about a shipping date or ask a tough technical question, you get a competent courteous answer. Outstanding customer service is a hallmark of Percom."

"Richard's making final changes to a disk controller which will allow Percom drives to be used with yet another computer. We're constantly developing and introducing new products that extend and enhance the value of Percom systems."



"Slipping a circuit board through the eye of a needle would be easier than slipping a cold solder joint past Beverly. These are four-drive LFD-400/800 disk system controllers she's inspecting."

Edited by Dennis Brisson

NEW PRODUCTS

PET Printer Interface

The TNW-1000 is an IEEE-488 bus serial interface for the Commodore PET/CBM. It provides an output-only serial port to interface to current-loop as well as standard RS-232 printers and other devices. It is designed to mount right to the back of the PET's cabinet but can be used with other IEEE-488-capable computers. An edgeboard connector is provided to allow daisy chaining of other devices on the IEEE bus with the TNW-1000.

You can select the baud rate from 110 to 9600 and switch-select automatic conversion from PET to ASCII character sets for either new- or old-style PETs, data word length and parity (8-bit words without parity or 7-bit words with even or odd parity) and IEEE bus address. Price is $129.

TNW Corporation, 3351 Hancock St., San Diego, CA 92110. Reader Service number 494.

Receive-Only Serial Printers

The Printerm Model 877 is an 8½ inch receive-only serial printer from Printer Terminal Communications Corp., 124 Tenth St., Ramona, CA 92065. It achieves high reliability through microprocessor control and low parts count.

The unit is a bidirectional 9 × 7 dot matrix printer featuring a heavy gauge, hardened metal chassis and stainless steel drive screw and utilizes a cartridge ribbon to eliminate ribbon reversing mechanisms as well as to assure print quality. It prints 120 characters per second at 80 characters per line, 10 characters per inch, using the ASCII set of 96 characters. The internally contained paper roll is friction fed.

The 877 utilizes a serial EIA RS-232C interface, with asynchronous baud rates from 300 to 9600. Dimensions are 18 × 22 × 7.5 inches and weight is 25 lbs. Price is $999. Reader Service number 482.

PIA Card

Quality Research Co., PO Box 7202, Spokane, WA 99207, offers a convenient way to design a custom interface or construct a timer, EPROM burner or A/D converter for your SS-50 system. The 80-205 PIA card is a prototyping card designed to fit one 30-pin I/O slot. It is a double-sided, plated-through board with lines etched for a 6820 or 6821 PIA that requires no hard-wiring of all the data and control lines.

The PIA outputs—as well as the following bus lines: reset, R/W, Ø2 clock, all five baud lines, ± 12 V, ground and the onboard regulated + 5 V line—are brought to a user area. The NMI and IRQ lines are jumper selectable for both sides of the PIA.

Although there is no solder masking or component labels on the board, assembly is easy to do. The two pages of instructions contain assembly information, a component layout and schematic.

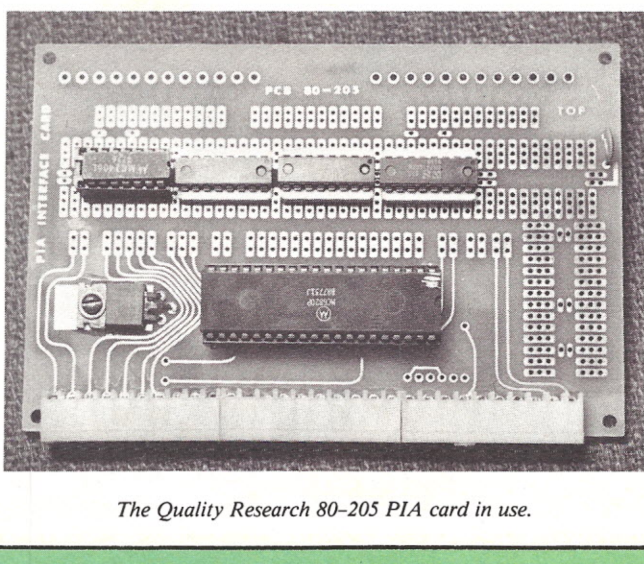
Shipment was prompt. I received my board and optional edge connectors 17 days after sending QRC the order. The 80-205 bareboard is $8.50, and the bottom edge connectors are $3, postpaid. Compared with the $14.95 Percom blank board and the Microworks $24.95 universal I/O board, the QRC product seems to be a better bargain if you are using a PIA. Reader Service number 475.

Dennis Doonan
Racine, WI 53403

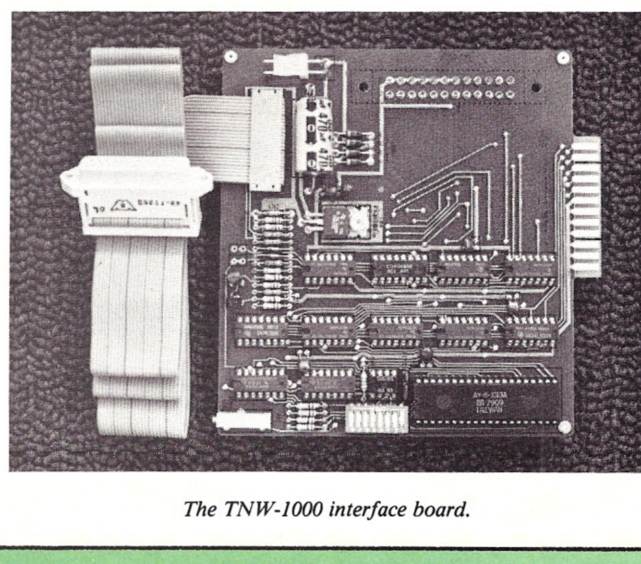
Lowercase Conversion Kit

Now you can have lowercase letters on your Centronics 779 or Radio Shack Line Printer I with the Conversion Kit I from Service Technologies, 3844 Spring St., Box 1426, Nashua, NH 03061. It allows a full 96 ASCII-character set with your choice of a zero with or without the slash.

This easy-to-install modification takes less than five minutes and only requires the use of a screwdriver to take off the back of the printer, remove one IC, plug in a small PC circuit board in its place and connect two jumper wires. There are no traces to cut nor connections to solder that will void your warranty. You can return the printer to its normal uppercase-only status by simply removing the kit and replacing the original IC. Price is $124.95. Reader Service number 493.



The Quality Research 80-205 PIA card in use.



The TNW-1000 interface board.

Pascal Terminal from ACI.

Pascal Terminal

The ACI Pascal video terminal is an optimized 12-inch CRT (24 lines × 80 characters) for use with the UCSD Pascal operating system or other applications requiring similar video terminal capabilities. It incorporates a separate ETX key along with standard UCSD Pascal X-Y cursor addressing. Home is defined as the upper left-hand corner.

The ACI Pascal Terminal provides a standard upper-lowercase 96 ASCII-character set. It accommodates several international language character displays by internal switch changes (no optional ROM required). Cursor display is switch selectable for either steady or blinking underline or block. The terminal also incorporates provisions to display 32 control characters. Characters are formed in a 5 × 8 pattern within a 7 × 10 dot matrix with descending cursor.

Associated Computer Industries, Inc., 17751 Sky Park East, Suite G, Irvine, CA 92714. Reader Service number 477.

Graphics and Scientific System

The AVTEK 1800 Graphics and Scientific System is a new microcomputer system that includes a 256 × 512 multigraphics terminal, 64K memory, dual 8-inch floppy-disk drives and floating-point processor. Hardware options include double-density drives, an X-Y plotter, a word-processor printer with scientific character wheel and a 132-column printer.

System software includes a screen-oriented editor, a word-processing package (with features for creating scientific papers), a communications package and both subroutine and stand-alone packages for graphics and scientific applications. Programs may be written in FORTRAN or assembler, with other languages available as options. AVTEK's new CP/M-based operating system provides substantial increases in speed and disk-storage capacity.

National Computer Communications Corp., 260 West Ave., Stamford, CT 06902. Reader Service number 485.

Apple II Printer

The Silentype thermal printer is an advanced text and graphics printer for the Apple II computer. With this low-cost peripheral you can print on paper copies of anything that the computer can display on a video monitor or television screen. Printing modes, formats and intensities are all variables that are under your control.

As its name suggests, the Silentype is completely silent when printing a line of text or graphics and makes only a slight sound when feeding paper to start a new line. It uses a 7-dot thick-film printing element to produce 5 × 7 dot characters and graphics on standard thermal paper. It prints up to 80 characters per line on 8½-inch-wide roll-fed paper. Price is $595.

Apple Computer, Inc., 10260 Bandley Drive, Cupertino, CA 95051. Reader Service number 497.

Head Cleaning Diskette

Scotch Head Cleaning Diskettes.

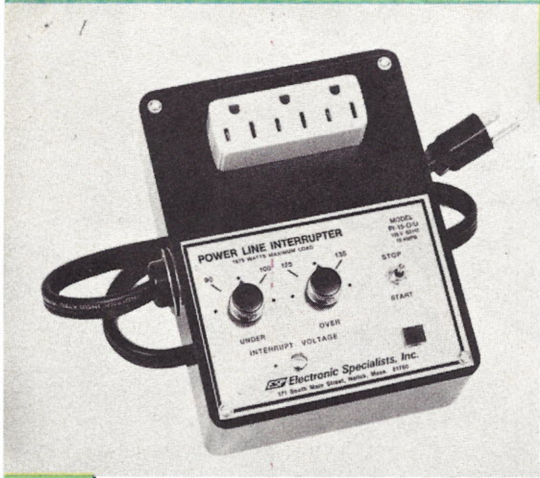
Cleaning kits for diskette heads to help cut down on system interruptions and losses of data are available from 3M, Dept. DR80-1, Box 33600, St. Paul, MN 55133. Scotch Head Cleaning Diskettes use a unique wet-and-dry method by which a proprietary cleaning solution is applied to the porous cleaning fabric in the diskette envelope. Then run the cleaning diskette for 30 seconds. Another portion of the diskette wipes the head dry, and the system can be used immediately. You can clean two-sided systems with the same technique.

Each kit contains two diskettes (specify either 8 inch or 5½ inch) and sufficient solution for 30 cleanings (15 per diskette). Price is $30. Reader Service number 480.

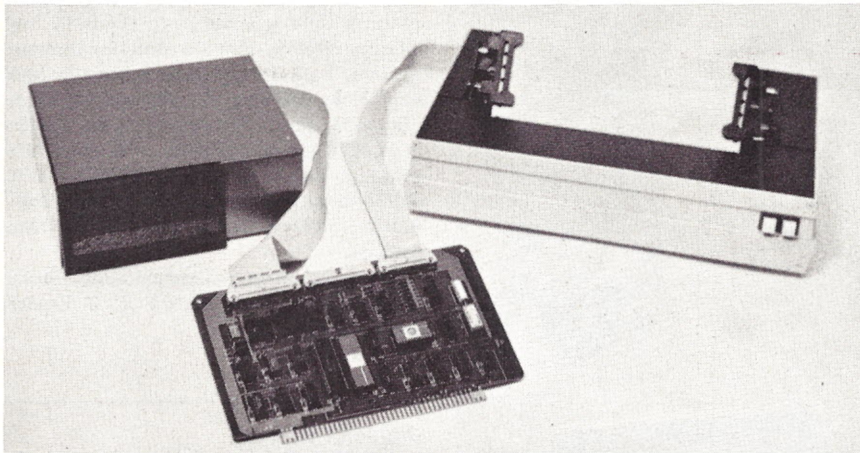
Power Line Interrupter

Now you needn't worry if the ac line voltage is disrupted or exceeds user-selectable limits. The Power Line Interrupter, with front-panel controls, disconnects power from the controlled apparatus and provides under/over

The AVTEK 1800 system.



ESP Power Line Interrupter.



The FP-950 Drive Controller.

voltage interrupt level selection and power reset. Other features include integral spike/surge suppression and response delay to prevent false interrupts.

Intended for specialized microcomputer applications where equipment is subject to periods of unattended operation, the interrupter will provide safety and protection for equipment as well as personnel. It connects to the ac line with a standard three-prong plug and can accommodate a 15 Amp resistive load or a 10 Amp inductive load. It is available in two models—over and under voltage ($142.95) or under voltage only ($127.95).

Electronic Specialists, Inc., 171 South Main Street, Natick, MA 01760. Reader Service number 479.

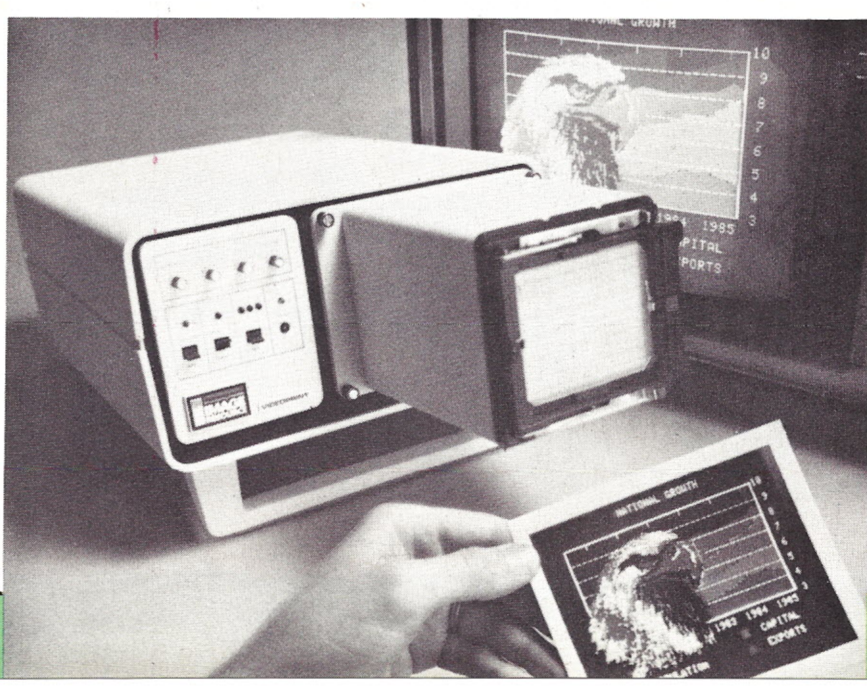
Full-Color Videoprint

Economical, full-color hard-copy reproductions of computer graphics are now available with the introduction of the new Videoprint Systems from Image Resource, 2260 Townsgate Rd., Westlake Village, CA 91361.

The Videoprint 3000 series ($2990 to $3550) is used as a hard-copy computer peripheral for the educational, small business and personal computer graphics markets. The Videoprint 5000 series ($5950) is a higher-performance system, equipped with more flexibility for use by the industrial and commercial computer graphics user.

Both systems are self-contained and fully automatic for minimizing optical distortion, with color, brightness and exposure adjustments under microprocessor control. Both the 3000 and 5000 series produce 4 × 5 inch hard-copy prints in seconds. Also available are Polaroid SX-70 and 35 mm sizes. The 5000 series has optional controls for unattended operation. Reader Service number 484.

The Videoprint produces accurate hard-copy records of computer graphics.



AIM-65 Floppy Disk Controller

The FP-950 Controller Module for the AIM-65 microcomputer controls up to four two-sided floppy heads (eight heads). It handles either single- or double-density formats and can be programmed for 128, 256, 512 or 1024 bytes/sector. The module can control 35, 40 or 77 tracks/side and features overlapped seek capability and independent motor control.

The new Advance Disk Operating System (ADOS) uses a new concept in the file structure which results in quick deletion and compression of files. It allows manipulation of I/O files in buffered sector or block transfer modes up to 63K. Files can be read/write from the editor, monitor, assembler or BASIC language, besides the normal commands for the manipulation of files. The FP-950 is $475. Single-sided drive with power supply and case is $375; double-sided drive is $460.

Applied Business Computer Co., 707G S. State College Blvd., Fullerton, CA 92631. Reader Service number 489.

Apple II Light Pen

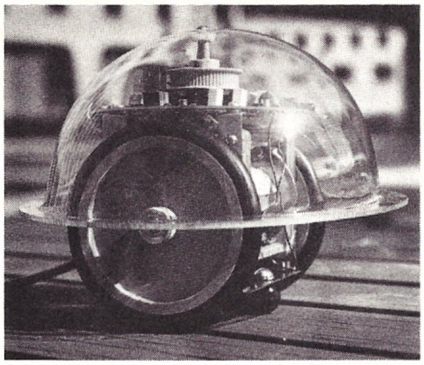
The Lipson Light Pen for the Apple II utilizes a cadmium selenide cell to detect and measure varying intensities of light. It is packaged with 12 BASIC programs on cassette tape, a 48-page manual, all the necessary cable and a connector to PDL(0) on the Apple II.

The manual instructs the user in the methods of detecting and measuring light. It includes demonstration programs (six in Integer BASIC, six in Applesoft) designed to be incorporated into the user's programs, along with tutorial comments and suggestions. Hi-res graphics, sound and color are implemented in the demonstration programs, and the user is encouraged to incorporate the demo routines into programs of his own design. Price is $24.95.

Aresco, Inc., PO Box 1142, Columbia, MD 21044. Reader Service number 487.

Turtle Interface

The TST-1 is a new interface that plugs into any standard 110 volt wall socket, into any standard serial line and then into the Terrapin Turtle, thus making Turtles completely pluggable for TRS-80, Apple or DEC. The TST-1 provides the Turtle with the parallel interface it needs as well as the necessary 18 volts dc (at 1.5



The Terrapin Turtle.

Amps) of power that it takes to run one of the small Turtle robots. TST-1s may be used to hook up computers directly or may be used for remote input/output. You can also hook up more than one Turtle to a computer, terminal or modem.

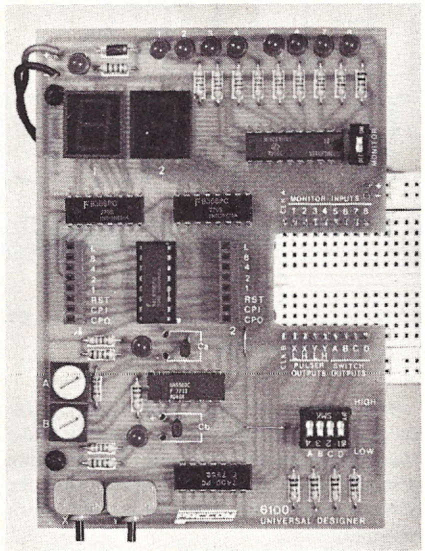
The TST-1 has a settable baud rate to allow it to operate at anywhere from 110 to 4800 baud. Price is $150, or $125 with a Turtle.

Terrapin, Inc., 678 Massachusetts Ave., Cambridge, MA 02139. Reader Service number 495.

Memory for the H11

The CI-1103 memory module is designed specifically for the Heathkit/Digital H11 computer, LSI 11/2 and PDP 11/03 microcomputers. The new memory features an 8K × 16 dual width board using 200 ns 4027 4K × 1 dynamic memory devices or a 32K × 16 dual width board using 200 ns 4116 16K dynamic memory devices. The unit simply plugs directly into the computers.

The CI-1103 is available with either onboard distributed refresh or external refresh control logic. Data access time is 300 ns, and cycle time is 525 ns. Onboard memory select is available in 2K increments up to 128K words of memory. The 8.44 × 5.187 inch board is available with battery backup capability. Power consumption is under 7 Watts. Price for the 8K × 16 is $390; the 32K × 16 costs $750.



The Paccom 6100 Universal Designer.

Chrislin Industries, Inc., 31352 Via Colinas #102, Westlake Village, CA 91361. Reader Service number 476.

Breadboarding Aid

The Paccom 6100 Universal Designer is an indispensable aid to breadboarding digital ICs from Paccom, 14905 N.E. 30th St., Redmond, WA 98052. By plugging directly into and utilizing Global Specialties or AP breadboarding strips, the 6100 creates a complete breadboarding system providing the most often used inputs and outputs for digital circuit design.

This design accessory features two bounceless push buttons, two seven-segment readouts with binary-coded decimal (BCD) input, four switch outputs, eight LED logic monitors, two variable clock generators, two decade counters and 5 volt supply pins for utilization of up to two breadboard power buses with the main breadboard design strip, for example, one Global Specialties QT-59S and two QT-59Bs. It comes with a schematic diagram, instruction manual and two demonstration experiments: a two-digit counter and a reaction timer. Reader Service number 498.

North Star hard disk drive system.

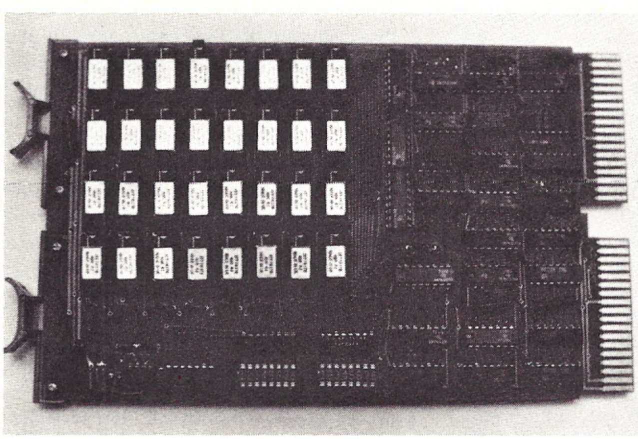
Hard Disk for Horizon

North Star Computers, Inc., 1440 Fourth St., Berkeley, CA 94710, has announced a new Winchester-type 18 MB hard-disk enhancement for its Horizon computers. To back up the information stored on the disk, the information that is modified each day is backed up on floppy disk, on a sector-by-sector basis. This backup and recovery system provides a convenient and inexpensive way to save and restore vital information stored on a hard disk. This system utilizes the Horizon's integral 5¼ inch floppy-disk drives, and data is saved on diskettes only if it has been changed since a previous backup.

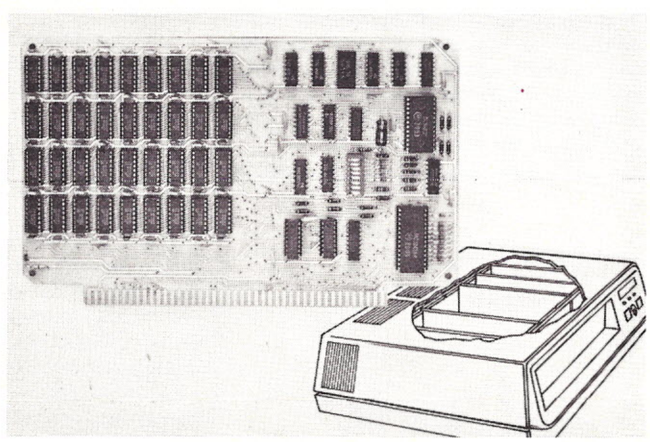
You can add up to four 18 MB Winchester-type disks for a total system capacity of 72 MB. Average access time is 78 ms. Reader Service number 496.

Exorciser Memory

The M64EX is a 64K dynamic RAM micromodule for the Motorola Exorciser and other Exorciser-bus-compatible systems. It features



The CI-1103 memory module.

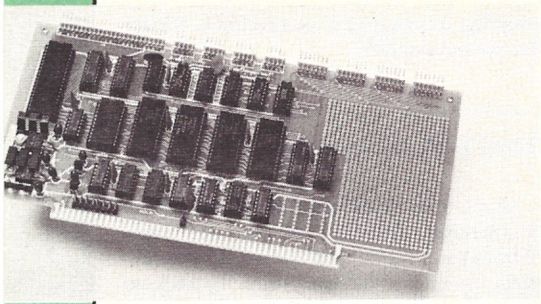


Percom Data's M64EX.

transparent refresh and optional parity check. It also includes an exclusive address translator circuit that accommodates program-controlled memory allocation, which is useful in applications such as multitasking.

The M64EX permits any combination of 4K blocks of RAM—within the upper 32K bytes of memory space—to be enabled or disabled with an onboard DIP switch. This enable-disable capability may be used with the address translator to implement functions such as write protection of program-selected memory blocks. The 64K RAM card is $875; 48K is $795. Parity check is $150 extra.

Percom Data Company, 211 N. Kirby, Garland, TX 75042. Reader Service number 488.



Micromation's multi-user S-100 board.

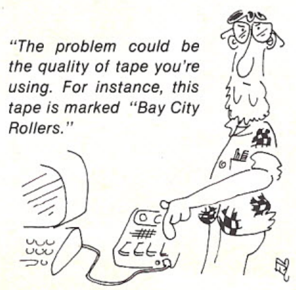
Multi-User S-100 Board

Micromation, 1620 Montgomery St., San Francisco, CA 94111, announces a multi-user S-100 board that features four RS-232 serial ports with full handshaking capability, three programmable timers, two bus-driving parallel output ports, three parallel input ports with handshake capability, plus a wire-wrap area for custom circuitry.

The four serial RS-232 I/O ports employ individual switch-selectable baud rates to 9600 baud. The four ports use 8551 UARTs to deliver full handshaking and full interrupt support. All UARTs and timers can be interrupt-driven for fast system operation.

The board includes an 8253 programmable interval timer, configured as a real-time clock for generating periodic system interrupts. Featuring software-selectable timing, this timer allows you to time events, print the date and time on reports, display the date, minute and hour on the screen and access the time via program software. Reader Service number 491.

"The problem could be the quality of tape you're using. For instance, this tape is marked "Bay City Rollers."



LETTERS TO THE EDITOR

France . . . er . . . America . . . er . . . New Zealand

With reference to your editorial notes (p. 28, October 1979; p. 2 and p. 46, June 1978): While your admonition about incorrect nomenclature is appreciated (i.e., Baudot's name when the Murray code is intended), your patriotism shows excess.

In fact, Donald Murray was a New Zealander. Born into a North Island farming community, he was of Scottish descent. He graduated BA from Auckland University in 1890, moved to Australia, where he worked as a journalist, and subsequently took an MA from Sydney University. His major subject was logic, particularly symbolic logic. In his articles in the *Telegraph and Telephone Journal* in 1914 and 1915, Murray remarked that he "knew little of machine technology as such but grasped the underlying principles required for an efficient system." Having these, "what followed was no more than a technological problem."

The Emile Baudot telegraph system using time division multiplex (TDM), to which Murray acknowledges his indebtedness, comprised three parts:

- Some printing mechanisms of the (American) Hughes system.
- Part of the Morse multiplex in which synchronously rotating distributors divided the line between two, three or four printing instruments.
- The five-unit code, which had been devised by Gauss and Weber.

The appellation "Baudot Code" is proper only in that the Gauss/Weber code is an adjunct of the Baudot Multiplex Telegraph System. This allocation of code elements is the International Alphabet No. 1. To Baudot goes the credit of being one of the pioneer "systems engineers."

The resemblance between the Baudot and Murray multiplex systems is restricted to the use of synchronously run distributors and a five-unit code. The disparity between the Baudot and Murray codes arises from the difference between the type of keyboard used.

The Baudot transmitter had five keys similar to those of a piano keyboard. Of these, the rightmost three were operated by the right hand (to give, in modern terms, the three least-significant binary bits), while the left hand operated the two most-significant bits. The allocation of combinations of the five keys was arranged to divide finger movement evenly between left and right hands and to minimize it for characters having the greatest frequency of occurrence.

The Murray transmitter used a typewriter keyboard so that the operator had to press only one key for each character. The allocation of code elements was such that characters occurring with the greatest frequency minimized both the keying operation and wear on the machine parts.

By removing the operator strain and by the inclusion of a tape-loop storage system, the Murray system operated at 40–45 wpm, as against the theoretical 30 and practical 20 wpm of the Baudot. It was in 1898 that Murray took his machine, then in the form of paper-tape control of printers' linotypes, to New York. There it was acquired by the Postal Telegraph-Cable Co., with the object of converting it to a printing telegraph. At this time, Murray took out patents for the future Murray Multiplex Telegraph. After developing this system into an automatic page-printing telegraph, he took it to England, in 1901, where it was taken up by the British Post Office.

From the UK, the Murray Multiplex Telegraph spread rapidly to all countries *except* the USA, which for four years maintained a tariff barrier against it to protect its own telegraph companies. Notwithstanding that, with some reservations, he had sold his American patents to Western Union, for whom the Murray telegraph equipment was manufactured by the Markrum Company.

As you say, Mr. Green, the credit for the common five-unit code (International Alphabet No. 2) does not go to one of those "confounded Frenchmen." Neither, however, does it go to a confused American, as you claim, but to the New Zealander, Donald Murray.

T. J. Seed
University of Canterbury
New Zealand

Computers in Education

Many educational fads have paused in the classroom en route to the storage room; only a few have earned a permanent place in the teacher's array of aids.

The current stars are classroom computers. I am a promoter of their increasing popularity, but the testimonials are as overblown as any touting previous newcomers to the classroom.

Frank Derfler made a start to defining a strategy to encourage newcomers to computers in educational applications (May 1980, p. 150). His approach is internally consistent, but I disagree with his premise that computer-assisted instruction (CAI) is the place to start. CAI is a valid approach to educational computing, but a poor introduction to teachers and programmers for at least two reasons. (1) No matter how transparent the programming, entry of

data consumes time busy teachers can't spare. This effort is better made by full-time programmers serving a broader population. (2) CAI shares this area of education with other quite successful media, programmed texts, for example.

Mr. Derfler notes applications programmed so far are in mathematics predominantly. Instead, here is a possible alternative to Mr. Derfler's approach.

Accept that the microcomputer is most comfortable in the math room at present and proceed in two ways.

In other disciplines, define and develop flexible data bases to employ the increased capacities now becoming available and to reduce the amount of user time dedicated to information. We need a few Diderots to produce on-line encyclopaediae.

Concurrently, in mathematics, develop the programming formats to make the micro a flexible and convenient aid. Yes, aid. No more, no less. Like the calculator before it, the computer opens new territory to study, so its claim to a space in the math class should be easily established. (CAI is handicapped by its data dependence in this charting-the-unknown capacity.)

Whether other subjects can be similarly expanded remains to be seen. What will transfer well is the programming style developed in math applications. For what they're worth, my goals for a classroom program are:

- Explore concepts beyond the capacity of other media.
- Permit as many student-subject exchanges as possible.
- Meet the teacher at his or her own level, as user, computer beginner or experienced programmer. This implies structured programming and levels of documentation from suggested lessons to variable lists.

The exciting, and threatening, prospect is that we'll be teaching students concepts we don't grasp ourselves; we'll have to learn right along with them.

Dave Goforth
GO:FORTH Microcomputing
Prince Albert, Saskatchewan
Canada

NAVPROG Refinements

I have recently completed entering Leland Young's NAVPROG program, and subsequent corrections, from the February 1980 issue, and have run some tests using FAA charts in the Los Angeles area. The program is fine, and I commend the author for his fine work. However, there appears to be a serious error regarding the computations of the magnetic variations. I have made the following corrections.

For stored data, line 280 sends you to line 4000. The processing at line 4000 completes at 4120 or 4150 and returns to 500. The formula for conversion of "easterly" variation to negative is line 490. I suggest 4120 and 4150 read 490, not 500.

Line 4110 uses the magnetic variation and the degrees thereof to index a variable previously used as an input variable. This violates all I ever learned about programming. In line 4110, I have changed V$(Q) and V(Q) to V1$(Q) and V1(Q), respectively. The new variables need dimensioning, and the READ statement in line 210 needs similar changes. The old program merely swapped the first item for the second for the third, etc., and the magnetic courses were wrong in all cases.

Ronald I. Ruby
Tarzana, CA

The comments made by Mr. Ruby are true. The use of V$(I), V$(Q), V(I) and V(Q), as in line 4110, resulted from repeated efforts to reduce the program size as much as possible. NAVPROG will work with the variables as listed, but it is a good idea to make the suggested changes. I have since expanded NAVPROG, and in so doing I had to change the variables as proposed. My thanks to Mr. Ruby for his interest and his comments.

In the vein of "there's always one more bug," it is necessary to include the SGN(X) function in the statement found in line 70:

```
DEF FN S6(X) = INT(X*10 + SGN(X)*.5)/10
```

This subtle change is necessary to ensure the correct "rounded-off" result for either positive or negative values of X. Heath BASIC requires this change, but Applesoft does not. The difference comes in the way the two BASICs handle the INT(X) function. In Heath BASIC, for example, INT(−3.3) = −3, but in Applesoft the result is −4.

Leland D. Young
Tampa, FL

Two on the 14

Regarding Thomas Prewitt's article on the Heath H14 printer in the February 1980 issue: The article mentioned the light printing using the ribbon furnished by Heath. I would like to make a suggestion.

I replaced the nylon ribbon furnished by Heath with a cotton Teletype ribbon (Nu-Kote #B72). It is a half inch by 12 yards, and also fits Teleprinters 10, 11, 47, 75, 444 and Envoy; plus Olympia SM3 and U50 and Westrex. It has made a big change in my hard copy.

Howard T. Dempsey
Fresno, CA

Author Thomas A. Prewitt submitted the following response to a long letter concerning his article on the Heath H14 Line Printer.—Editors.

After several hours of printing, the H14 may refuse to perform a line feed. The most likely cause for this is simply paper fibers and punchouts from tractor holes. This material collects in the paper feed path and blocks off the phototransistor that monitors the movement of tractor holes in the left edge of the paper to detect paper jams. Correction is as simple as brushing or blowing the foreign material away.

There is a discrepancy between the description of the printhead fuse shown on the schematic and that which appears elsewhere in the manual. The correct part is a 6/10 Ampere slow-blow, not a 1 Ampere fast-blow as shown on the schematic and as a legend on the printed circuit board. Certain forms of circuit component failure can overheat the printhead without blowing the 1 Ampere fast-blow fuse. Printheads are expensive!

T. A. Prewitt, W9IJ
Kokomo, IN

PC Board for Power Supply

O.C. Stafford Electronics Service and Development Company, 427 South Benbow Road, Greensboro, NC 27401, is offering a PC board for the "Mighty Five-Way Power Supply" (April 1980, p. 216).

Part No.	MC-4-80
drilled	$5.60
undrilled	4.50
shipping	1.00

Don Walters
Ann Arbor, MI

Thanks

I continue to see letters from readers complaining about the lack of articles on micros other than the more popular models: TRS-80, PET, etc. My letter, however, is to thank you for "A Not-So-Fast Renumberer for OSI BASIC" by John W. Aughey, which appeared in the January 1980 issue.

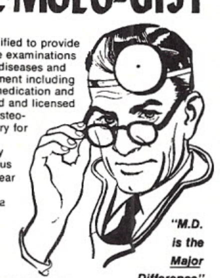
Peter Gibbs
Mississauga, Ontario
Canada

The following was sent anonymously to remind us that we made a mistake in titling Robert Bauman's article in the May 1980 issue "Physician: Automate Thyself."—Editors.

DIFFICULT TO SPELL . . . EASY TO UNDERSTAND!

OPH-THAL·MOL'O-GIST

A primary care physician qualified to provide comprehensive diagnostic eye examinations for both systemic and ocular diseases and the initiation of medical treatment including the prescribing of indicated medication and lenses. He is educated, trained and licensed as a Doctor of Medicine (or Osteopathy) and is the portal of entry for the public into medical care systems. His education usually includes 4 years of college, plus 4 years of medical school, 1 year of internship and 3-4 years of ophthalmology residency, for a total of 12-13 years of "basic training."



"M.D. is the Major Difference"

OP·TOM'E·TRIST

A limited practitioner, whose formal education (usually two years pre-optometry college classroom study, plus a 4 year college curriculum in optometry) limits him to testing for vision problems unrelated to disease. Optometrists test depth and color perception and the ability to focus and coordinate the eyes. When necessary, they prescribe and fit lenses. Most optometrists sell eyeglasses. Some are taking additional classroom training in an effort to expand their services into the practice of medicine. Ocular pharmacologists (M.D.'s) testify that classroom training is inadequate . . . and that this trend is a public health hazard.

PHYSICIANS EDUCATION NETWORK INC.

EXATRON STRINGY FLOPPY Owners Association Newsletter

Secretary, Fred Waters

Let's put a few ideas together concerning money and taxes. Everyone is aware of the tax revolt that is spreading throughout the land. It started here in California, and it kept on going. What can you as an individual do about it on your own behalf? We've been reading Wayne Green's editorials and publisher's remarks for a long time about producing some income to support your hobby. Last year Personal Computing printed two articles, one by Paul Snigier and another by Mark Battersby, about using your computer to establish a business and get the tax breaks. (For copies, call us.)

LIKE TO BE A BUSINESS?

How would you like to be able to support your computer activities financially–to the point where the costs of equipment, software, magazine subscriptions and the like can be balanced against income, not only that derived from your personal computer, but also that from other sources? Well, Uncle Sam and the IRS would like to help you pay for all this! You don't believe it? Order a Stringy Floppy–the fast, accurate, reliable, and inexpensive mass storage device for microcomputers–and join the bandwagon. We'll help you get started in your own business, with all the tax benefits. Read on!

First, a couple of thoughts. You may well ask, why is the Newsletter for the EXATRON STRINGY FLOPPY OWNERS ASSOCIATION–owners of the world's finest alternative mass storage subsystem for small systems–interested in this subject? Good question, and three possible answers. First, the present ESFOA Workshop Chairmen (there are now several hundred, all over the U.S.) are already in just the right situation to set themselves up this way. You chairmen are already acting as focal points for answering questions locally, for demonstrating the ESF to prospects who want to be shown, and for channelling orders back to Exatron, thereby generating income. Second, there are a lot of you ESF owners who haven't yet decided to act as Workshop Chairmen–your choice–and this might be your incentive. Third, there are a lot of you microcomputer owners who have not yet discovered the Exatron Stringy Floppy, along with all the remarkable benefits of an amazing owners association–full hardware and software support, enthusiastic mutual assistance, interchange of user techniques and ideas, and widespread common backing of this exciting micro-peripheral. Here's a way for those of you in this last group to acquire the ESF and defray all costs.

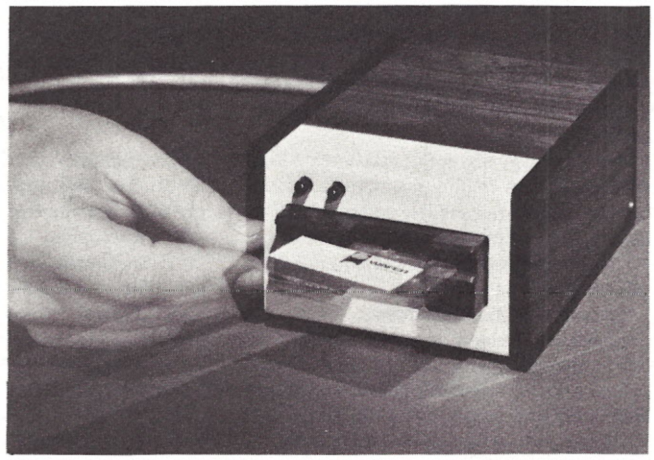
Another thought. We aren't tax accountants or tax attorneys –we just know how the IRS rules apply to us. Use the information and tips in this newsletter, get a copy of "Tax Guide For Small Business", and consult your own tax advisor.

HOW DO I QUALIFY?

There are several things you must do, and several others you should do, to establish yourself as a bona fide business. First, and most important, the profit motive must be present, and there must be evidence to support this. If you show a net profit in two out of five years, there is a presumption that you are engaged in business for profit. Even when this is not the case, you can demonstrate the intent by supporting activities.

Second, you must keep records. You must choose an accounting method, keep good records of all transactions, and retain copies of all supporting documents. To quote the IRS, "Two characteristic elements of a business are regularity of activities and transactions, and the production of an income."

Evidence of your own skills and expertise help show that the profit motive is present. If you have written a program and sold it to Instant Software, you prove your skills. If you own a Stringy Floppy and have demonstrated it to others in your city, and thereby generate orders and receive bonuses and commissions, you have proven your expertise in that aspect of your computer activities.



Call your Program Chairman to get a demonstration of the Exatron TRS-80 Stringy Floppy.

The time and effort you spend on your computer activities is backup for your status as a bona fide business. Even if you have another full-time occupation, you could easily be putting in 10 to 30 hours a week at your business, and this helps to show that you mean it.

If you avail yourself of the paid services of consultants, this again tends to establish your good intention to make a profit. Your cancelled checks for tax advice and technical consulting services are an important part of your records.

Well, here you are. You have set yourself up as a small business, and have in good faith met the standards and qualifications set by the IRS. Your next question would be

WHAT ARE THE BENEFITS?

At this point you have to make a big shift in perspective on the way the tax laws apply. You're probably familiar with the way they work in your case for individual returns, or joint returns for man and wife. For a business it's quite a bit different. To repeat, get a copy of the IRS "Tax Guide For Small Business" and get a tax adviser.

In general, there are credits, deductions, and depreciation. When you acquire capital equipment–which means practically all hardware, from your computer on down to ESF wafers for program and data storage–you qualify for an investment tax credit of 10% of the cost. There are qualifications to meet, and are other credits available, so check this one out.

Here are some of the items that can qualify as deductions. Software that you buy to operate your business. The cost of trade journals (your Microcomputing subscription), reference books, and education to maintain or improve skills required in your business. Business travel, transportation, and mileage. The costs associated with your attendance at trade shows, or your participation. Check it out!

INFORMATION & ORDERS

The ESF is assembled and tested at the factory, with a 30-day moneyback guarantee and a one-year full warranty. For fastest delivery, phone in your credit card or COD order using the toll-free line below.

Base price for the TRS-80 ESF, $249.50 (ask about the Starter Kit); for the S-100 ESF $289.50. The RS-232 compatible MICROSPONGE is $349.50 with substantial discounts for OEM quantities. Info packets at no charge; TRS-80 (only) users manuals are $3 for shipping and handling.

If you have any questions about these products, about Exatron or ESFOA please call the Hot Line. Address letters to ESFOA, 3559 Ryder St., Santa Clara, CA 95051.

HOT LINE	800-538-8559
WITHIN CALIFORNIA	408-737-7111

Stringy Floppy is a trademark of Exatron Corporation. ✓7

EXATRON STRINGY FLOPPY
WORKSHOP CHAIRMEN

NOTICE: The following is a list of ESF Workshop Chairmen (listed by zip code). If you would like a demonstration of the Stringy Floppy, or if you have any questions about the hardware or software pertaining to the ESF, please feel free to call on the chairman in your area. They will be happy to help you.

WORKSHOP CHAIRMEN (BY ZIP CODE)

T. De Man, Holland
John Parker, Port Alberni BC ( ) 723-5861
J.K. Welsh, Burnaby BC (604) 521-3275
Alec Utting, New Zealand (4) 786-524
Karin Golinski, D-6392 Neu-Anspach 1
R.W. Hall, San Carlos, CA (415) 347-9521
Rob Vervlost, Holland
Allan Stubbs, Prince George BC ( ) 563-6894
Roy Wallace, Aylmer, Quebec Canada J9H6B5 (819) 684-9087
A.P. McKinley, Saskatoon, SA (306) 343-8887
John Ragle, Amherst, MA 01002 (413) 253-3041
Jack O'Connell, Marlboro, MA 01752 (617) 481-2417
Austin McCollough, Chelmsford, MA 01824 (617) 256-0473
John Pyra, Westford, MA 01886 (617) 692-8070
Ian MacDonald, Bristol, RI 02809 (401) 253-8675
Dr. R.M. Knowles, Cumberland Foreside, ME 04110 (207) 773-7261
Daniel Haggerty, Bangor, ME 04401 945-5547
Wesley Day, Waterville, ME 04901 (207) 873-3955
Rev. Richard Noyes, Madison, ME 04950 (207) 696-5929
Michael Keller, Solon, ME 04979 (207) 643-2437
Thomas Penish, Bogata, NJ 07603 (201) 343-3828
Rev. Edward Collins, Schooley's Mountain, NJ 07870 (210) 876-4395
Ted Kingston, Strathmere, NJ 08248 (609) 263-2440
Major Donald Gober, APO NY 09052 06332-5135
Milton Thrasher, Rye, NY 10580 (914) 967-3322
John Sheil, Levittown, NY 11756 (516) 731-6467
W.C. Hitzmann, Newtonville, NY 12128 (518) 783-9495
Bill Firm, Kingston, NY 12401 (914) 383-1824
George Hadley, Paul Smith's, NY 12970 (518) 327-6413
Lew Daly, Fayetteville, NY 13066 (315) 637-8886
Richard McGarvey, Williamsville, NY 14221 (716) 634-3026
Robert Wynne, Niagra Falls, NY 14303 (716) 285-9391
Gary Wnyder, Rochester, NY 14607 (716) 461-1364
David MacAdam, Rochester, NY 14615 (716) 865-8917
Paul Sturpe, Corapolis, PA 15108 (412) 264-1492
Bill Dalesandry, Pittsburgh, PA 15243 (412) 221-1968
Steven Smith, Altoona, PA 16602 (814) 943-5947
Harold Kautz, Jr., Hershey, PA 17033 (717) 534-2642
Lon Culbertson, York, PA 17404 (717) 843-3425
William Gieske, Nazareth, PA 18064 (215) 837-0223
Carl Killar, Nanticoke, PA 18634 (717) 675-5377
Bruce McIntyre, Langhorne, PA 19047 (215) 322-1895
Richard Landy, Morrisville, PA 19067 (215) 736-0564
Robert Kelch, Glen Burnie, MD 21061 (301) 760-4793
Charles Sutton, Salisbury, MD 21801 (301) 546-0413
Richard Harrison, Warrenton, VA 22186 (703) 439-8109
William Schultze, Fredericksburg, VA 22401 (703) 786-8878
William Geer, Falmouth, VA 22401 (703) 371-2019
Stephen Moulton, Virginia Beach, VA 23454 (804) 486-8482
H. Ross Wiant, Grafton, VA 23692 (804) 898-8120
Archie Hubbard III, Bristol, VA 24201 (703) 669-3191
Lawrence Doyle, Eden, NC 27288 (919) 627-0359
William Cornwell, Cary, NC 27511 (919) 467-9156
Micalel Talbot, Raleigh, NC 27612 (919) 787-8082
Kenneth Hendricks, Charlotte, NC 28210 (704) 552-1630
Walter Chandler, Columbia, SC 29210 (803) 781-4822
Lawrence Elliott, Summerville, SC 29483 (803) 873-6765
F.B. Collum, Florence, SC 29501 (803) 662-0421
Herbert Alperin, Atlanta, GA 30342 (404) 231-2310
Owen Black, Augusta, CA 30907
Mark Few, Orlando, FL 32803 (305) 644-7464
J.T. Carlton, Satilite Beach, FL 32937 (305) 773-1590
Richard Nelson, Marathon, FL 33050 (305) 743-9195
Frank Lungarella, Miami Springs, FL 33166 (305) 887-4821
A.J. Catano, Sunmise, FL 33323 (305) 741-4325
R.W. Hopkins, Bradenton, FL 33505 (813) 748-7074
William Howes, Ft. Myers, FL 33907 (813) 482-0347
Bob Jones, Memphis, TN 38138 (901) 363-5469
R.L. Thomas, Jackson, MS 39212 (601) 373-4231
Sumner Loomis, Prairie Pt., MS 39353 (601) 726-5524
L.W. Stein, Independence, KY 41051 (606) 371-3242
W.H. Jenkins, Canal Winchester, OH 43110 (614) 837-2325
G.C. Hufnagle, Rickenbacker AFB, OH 43217 (614) 491-1101
L.R. Snowden, RAFB, OH 43217 (614) 497-0838
L.C. Graham III, Barnesville, OH 43713 (614) 425-3633
T. Mason, Cleveland, OH 44115 (216) 575-5647
Paul Schmid, Orrville, OH 44667 (216) 682-3596
C.V. Elrod, Mansfield, OH 44907 (419) 756-1334
Robert Morgan, Celina, OH 45822 (419) 586-7416
Don Heringhaus, Ottawa, OH 45875 (419) 523-5613
W.R. Gouge, Indianapolis, IN 46241 (317) 856-6339
D.L. Cook, South Bend, IN 46614 (219) 291-2888
D.M. Gropper, Newburgh, IN 47630 (812) 925-7142
R.M. Clark, Redford, MI 48240 (313) 535-5722
R.L. Harrison, Saginaw, MI 48603 (517) 793-0166
J.J. Fleming, Royal Oak, MI 48703 (313) 435-4487
Dennis Chaney, Owosso, MI 48867 (517) 723-1011
A.C. Lanphear, Kalamazoo , MI 49005 (313) 344-8277
Ranjit Diol, Dowagiac, MI 49047 (616) 424-5440
R. Greenwood, Lowell, MI 49331 (616) 897-8634
Harley Dyk, Grand Haven, MI 49417 (616) 846-2877
D.R. Meinke, Grand Rapids, MI 49507 (616) 245-9420
R.L. Knapp, Jr., Perry, IA 50220 (515) 465-5873
R.F. Gerald, Arden Hills, MN 55112 (612) 636-0549
D.L. Guerin, Minnetonka, MN 55343 (612) 935-5264
M. Haugrud, Grafton, ND 58237 (701) 352-3793
R.W. Foster, Lake Zurich, IL 60047 (312) 438-5223
Martin Berger, Skokie, IL 60076 (312) 674-3038
C.J. Opperman, Carol Stream, IL 60187 (312) 682-3870
Joel Natkin, Evanston, IL 60202 (312) 869-1660
G.P. Cocanour, Hinsdale, IL 60521 (312) 986-1059
H.S. Leopold, Hinsdale, IL 60521 (312) 325-6133
C.M. Klindworth, Plainfield, IL 60544 (815) 436-2205
James Spath, Chicago, IL 60613 (312) 353-2544
R.M. Kramer, Abingdon, IL 61410 (309) 462-2859
J.M. Delaney, Jr., Wood River, IL 62095 (618) 254-4571
D.L. Hiles, Beardstown, IL 62618 (217) 323-3963
Lynn Nelson, O'Fallon, MO 63366 (314) 272-1153
James Runyan, Rolla, MO 65401 (314) 364-7306
Ed Forest, Prairie Village, KS 66208 (913) 648-1515
C.L. Perrin, Wichita, KS 67218 (316) 685-0528
Val Lindholm, Great Bend, KS 67530 (316) 793-9319
Jeffry Bretz, Omaha, NE 68134 (402) 397-0134
R.K. Lahm, Holdrege, NE 68949 (308) 995-4071
Herb Petermann, Oklahoma City, OK 73135 (405) 672-3532
Randy Wright, Plano, TX 75074 (214) 596-7000
Dr. Murlon Dye, Commerce, TX 75428 (214) 886-6434
Donald Michel, Arlington, TX 76016 (817) 469-1521
W.L. Lonnon, Decatur, TX 76234 (817) 627-3838
Kent Zakrzewski, Houston, TX 77007 (713) 448-6374
George Lyda, New Baden, TX 77870 (713) 828-4523
Stephen Fisch, Corpus Christi, TX 78413 (512) 991-4584
R.B. Sexton, Austin, TX 78757 (512) 452-6131
John Dierdorf, Austin, TX 78759 (512) 345-3912
Gary Crouse, Broomfield, CO 80020 (303) 466-5863
Eric Keener, Broomfield, CO 80020 (303) 422-2775
Richard Laubhan, Colorado Springs, CO 80909 (303) 632-7580
D.E. Murray, Ogden, UT 84404 (801) 782-8042
John Crutcher, Scottsdale, AZ 85254 (602) 948-2911
Bob Rankin, Tempe, AZ 85282 (602) 967-7677
Earles McCaul, Yuma, AZ 85364 (602) 344-3875
Thomas Mattingly, Las Vegas, NV 89107 (702) 870-1811
John Olip, Chatsworth, CA 91311 (213) 998-5445
Eva Gordon, Chatsworth, CA 91311 (213) 998-5445
R.D. Sullivan, Upland, CA 91786 (714) 985-5273
F.M. Harris, San Diego, CA 92116 (714) 281-9286
Jim Larman, Westminster, CA 92683 (714) 897-0577
Don Haskell, Visalia, CA 93277 (209) 732-1553
Hubert Wood, Bakersfield, CA 93308 (805) 399-8252
Bill Morley, Morro Bay, CA 93442 (805) 772-3240
Wm. Marks, Monterey, CA 93940 (408) 372-6975
Burham Bill, Redwood City, CA 94062 (415) 365-0760
A.C. Vincent, Napa, CA 94558 (707) 224-5144
R.H. Edmonds, Oakland, CA 94612 (415) 465-9180
James Ross, Richmond, CA 94801 (415) 232-0553
G.K. Johnson, Petaluma, CA 94952 (415) 442-2359
Dustin Leer, Sausalito, CA 94965 (415) 332-2557
James Ferguson, San Jose, CA 95132 (408) 946-1265
Lars Benson, Santa Rosa, CA 95401 (707) 523-1782
Richard Mack, Santa Rosa, CA 95401 (707) 527-4261
J.J. Waters, Santa Rosa, CA 95405 (707) 539-2595
V.A. Jupe, Carlotta, CA 95528 (707) 768-3833
E.R. Hubbard, Shingle Springs, CA 95682 (916) 677-5440
Paul LeFevre, Sacramento, CA 95825 (916) 483-4760
A.J. Kiep Jr., Etna, CA 96027 (916) 467-3255
G.A. Olayan, Kailua, HI 96734 (808) 261-2228
Andrew Voza, American Samoa 96799
Leo Folz, Salem, OR 97302
Ken Ernst, Salem, OR 97303 (503) 393-1173
James Shunn, Klamath Falls, OR 97601 (503) 882-2023
Rick Winter, Pendleton, OR 97801 (503) 276-3611
J.A. Records, LaGrande, OR 97850 (503) 963-0437
David Graham, Poulsbo, WA 98370 (206) 779-5105
Ralph Baker, Long Beach, WA 98631 (503) 642-3136
Jerry Starzinski, Yakima, WA 98903 (509) 452-2540
Allen Greenburg, Spokane, WA 99201 (509) 624-1111

(ADVERTISEMENT)

Get Your Pet on The IEEE 488 Bus

This 3-part odyssey takes you along route 488. The first stop is here . . . tickets, please.

Gregory Yob
Box 354
Palo Alto, CA 94301

Perhaps the most obscure Commodore PET feature is its IEEE 488 (or HPIB or GPIB) interface. This three-part article describes the rudiments of the 488 bus and how to use your PET to communicate with instruments having the 488 interface. Several working examples with Hewlett-Packard equipment are shown. (HP lent me several 488-compatible instruments to prepare this article.)

If you just want your PET to talk to that costly instrument on your bench, skip this month's installment and start next time with part 2. The first two parts of this article will sketch the prerequisites and give you enough information to track down bugs on your own.

What's a 488 Bus?

In 1972, engineers—some with Hewlett-Packard—proposed a method of joining many instruments in a standardized way to help automate lab and test measurements. This resulted in the IEEE Standard 488-1975, which describes how to connect as many as 15 instruments on the same cable.

HP and several other laboratory-instrument manufacturers then offered the IEEE 488 scheme as an option. Presently, several hundred instruments have the 488 capability; Commodore used to provide a 5-page list of these. The PET was later designed with the instrumentation and control market in mind, so the IEEE 488 interface was put into the PET.

Before the introduction of the PET, instruments capable of controlling the 488 bus cost several thousand dollars. Now the PET often costs less than the instruments it controls. Some 488 manufacturers have trouble adjusting to this—their customers balk at the idea of purchasing an $800 microcomputer to control a $30,000 instrument!

Now one connector joins the PET to many peripherals. You don't need a separate interface and connector for each new gadget. Commodore's printer and disk are designed to use the PET's 488 interface.

Physical Aspects

A PET and a 488-compatible device have different connectors. Your first project is to wire a cable to tie the two machines together.

Fig. 1 shows the location of the IEEE 488 connector on the back of the PET, and Fig. 2 describes the pins and connectors used for the PET and the IEEE 488. I used a 20-conductor ribbon cable and tied the

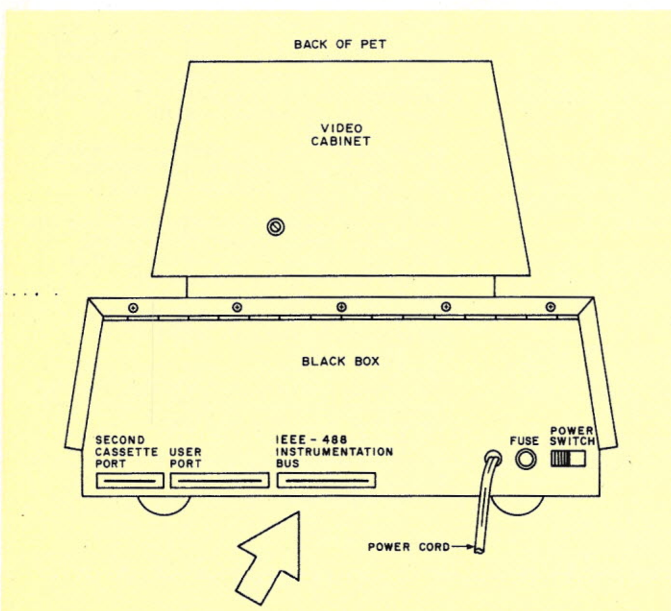


Fig. 1. Location of PET IEEE 488 port on the back of the PET next to the power switch and fuse.

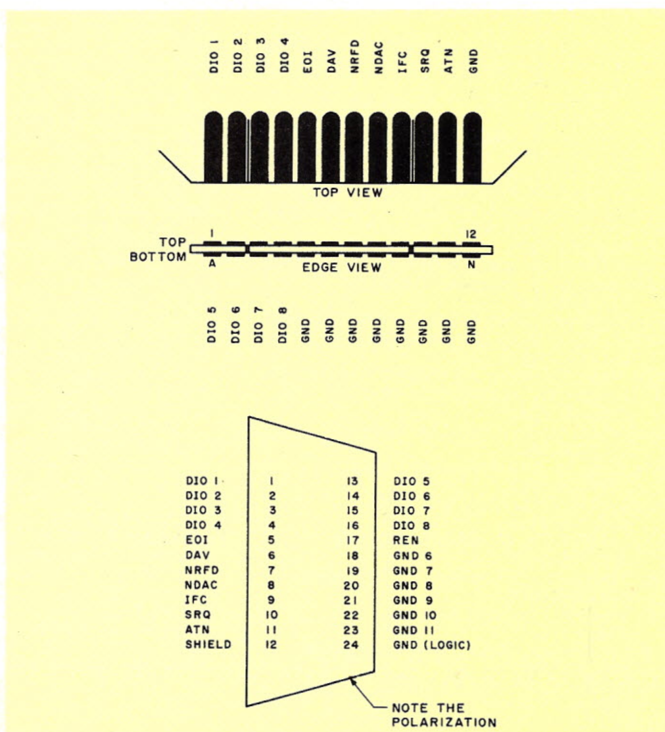


Fig. 2. Pin-outs and connectors for the IEEE 488.

grounds together into the four lines left over after I connected the signal wires.

When making the cable, bear in mind that there are strict limits to cable lengths:

1. The maximum distance between two devices is 5 meters.
2. The longest distance from one end of your setup to the other is 20 meters.
3. A maximum of 15 devices, including the PET, can be hooked together.

It is also wise to avoid electrically noisy areas; don't drape your IEEE 488 cable over your TV set.

If more than one device is connected to the 488, you must use extension cables. HP has cables for about $50. If you want to make your own, consult the two configurations in Fig. 3. The 488 instruments always have a female connector, so have an excess of male connectors on your cables.

Electrically, the 488 bus works on an active-low principle. Fig. 4 shows a circuit similar to a 488 bus line. When all the switches are open, the voltmeter will show 5 volts, which is the false state (or 0) for the line. If any of the switches are closed, the line is grounded, and the voltmeter shows zero volts, or the true state.

This peculiar arrangement permits several devices to be connected to the same line. If any one of them has a switch closed, the line is true. Devices frequently operate at different speeds, and when each device is ready, it opens its switch. However, the line remains true (low) until the slowest device opens its switch.

IEEE Blinkin Lites Display

It is always convenient to have a display and switches to perform a front panel function when you debug interfaces. I built a box, which I call the 488 Blinkin Lites, to display the states of each of the IEEE 488 lines and some switches to force lines low if needed. Fig. 5 shows the circuit, and Fig. 6 is a sketch of my box.

Each line is pulled up to +5 volts with a 10k resistor—the high value was chosen to minimize the load on the 488 bus. The switches can override any line when they are closed to ground. Though the PET doesn't use all the IEEE 488 lines, future machines will—so I put them all in my box.

If you build this box, don't use the PET's +5 volts from the tape port—the LEDs draw 170 mA, which is too much for the PET. Provide a connector to the PET's IEEE port and a male and female IEEE connector. This lets you interpose the IEEE Blinkin Lites between the PET and an instrument.

I mounted a 5×7 inch perfboard with 0.10 inch holes into a standard breadboard box and placed a label near each switch/LED combination to identify the IEEE lines. The three ICs are the 7404s used to drive the LEDs. The cable leads to a homemade junction with a PET connector and IEEE male and female connectors. A mini phono jack connects to a separate +5 volt supply (see Fig. 6).

When you plug in the IEEE Blinkin Lites, the LEDs will show the state of the lines—an LED that is off indicates a low line, which is true; an on LED indicates high, which is false.

The IEEE 488 Lines

The IEEE 488 is composed of 16 lines. Eight are for transfer of data, five are for bus management and three are for handshaking. The eight data lines are labeled DIO1 through DIO8, with the most significant bit (MSB) on DIO8. The 488 bus can transfer one byte at a time and is sometimes called byte-parallel.

The five bus-management lines in various combinations and sequences provide many bus facilities, most of which are rarely used:

EOI—End of Message. When a group of bytes is sent via the DIO lines, EOI is made true on the last byte to indicate that the message is completed. This is optional, and many instruments send the ASCII characters CR and LF as data instead. Check your instrument's manual.

IFC—Interface Clear. When this line is true, all instruments disconnect to a defined state. (This usually is unaddressed and untalked.) When you turn on the PET, IFC is true for about 100 ms. If the PET is reset, IFC will again be true.

SRQ—Service Request. This permits an instrument to signal that it needs attention . . . and the device in charge of the bus must find out what it needs. The PET has this line as an input, but it takes some programming effort to use SRQ; most instruments don't use SRQ.

REN—Remote Enable. Most IEEE instruments have front panels that permit stand-alone operation—that is, they work as ordinary instruments when the 488 bus isn't connected. REN lets the instrument disconnect from the bus and be controlled from its front panel.

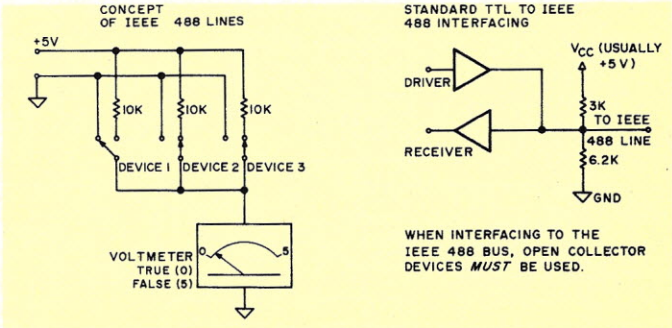


Fig. 4. IEEE 488 equivalent circuits. The lower circuit is the standard method of connecting TTL logic to the 488 bus. The driver must be an open collector and able to sink at least 48 mA at .4 volts and source 5.2 mA at 2.4 or more volts. The PET uses MC3446P bidirectional line interface ICs for this function.

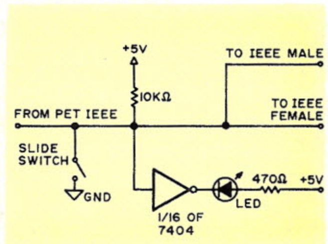


Fig. 5. IEEE "Blinkin Lites" circuit. Each IEEE line uses one copy of this circuit.

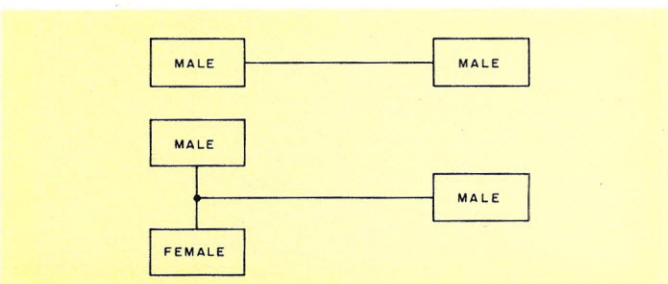


Fig. 3. Convenient cable configurations for the IEEE 488 bus.

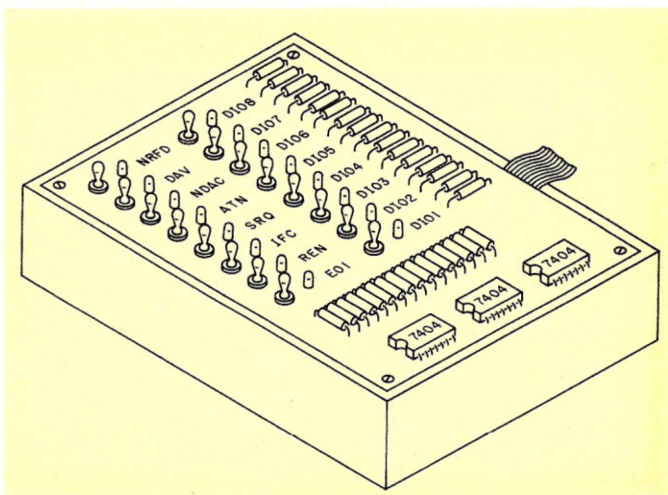


Fig. 6. Sketch of the "Blinkin Lites."

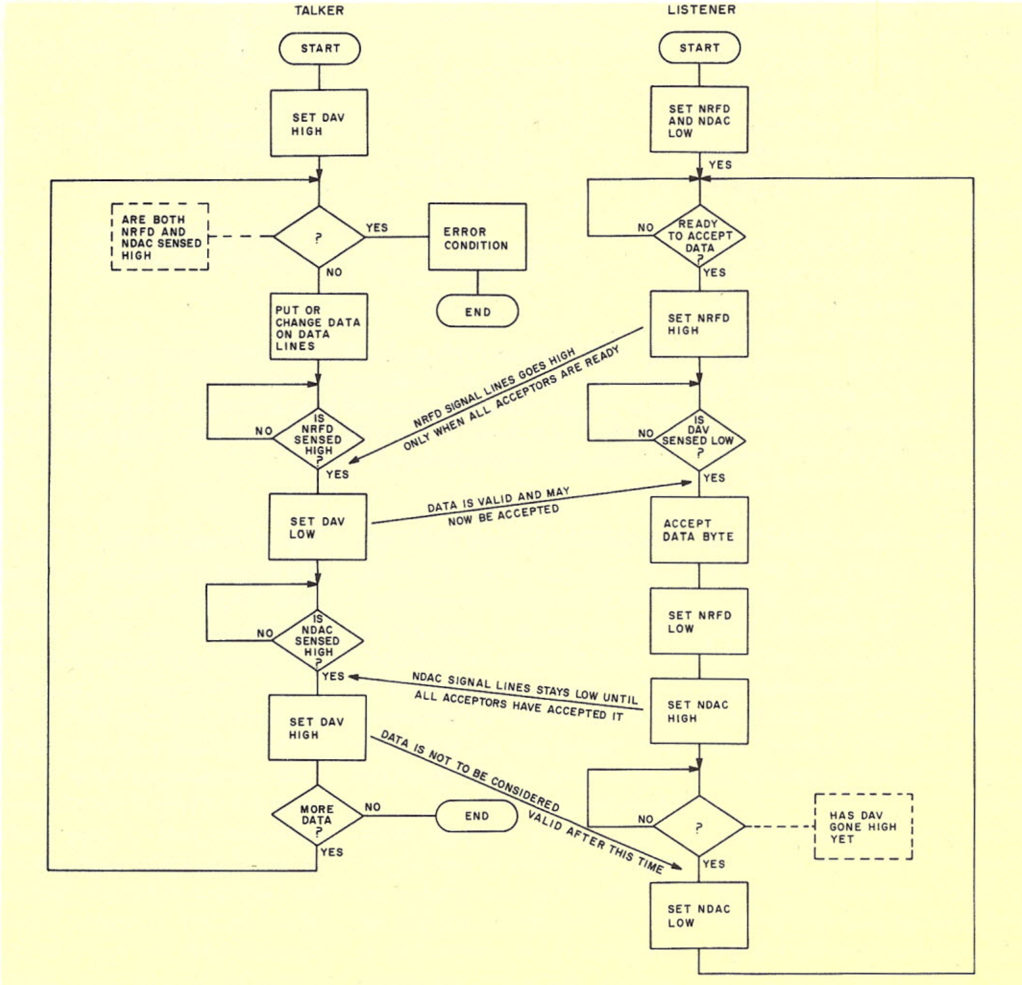


Fig. 7. The IEEE 488 handshake reproduced from Electronics, *Nov. 14, 1974, p. 98, as reproduced in HP part #5952-0058.*

The PET's REN line is always grounded.

ATN—Attention. This is the most relevant line for this article. It tells the device whether to regard the data on the DIO lines as a command or as data. When ATN is true, the byte on the DIO lines is a command. When ATN is false, DIO is seen as data.

The three handshake lines are used to pass bytes on the DIO lines. When a byte is transferred, the slow devices will keep one or more of the handshake lines true until they are finished. This ensures that data is passed at the speed of the slowest device and isn't lost.

The handshake lines are:

DAV (Data Valid)—When this line is true, the data on the DIO lines is correct and the receiving instruments can pick up the byte.

NRFD (Not Ready For Data)—When a receiving device is busy or is still processing prior data, it will make NRFD true, which stops data transfers.

NDAC (Not Data Accepted)—When the data is on the DIO lines, the receiving devices keep NDAC true until all of them have read the data byte. Note that the handshake lines don't care whether the data is a command or not; every byte of data or command has to undergo the handshake sequence.

The Handshake

For data transfer, one device is the "talker," which provides the data or commands for transfer. The recipients, or "listeners," pick up the data, and more than one device may listen at the same time. The handshake specifies exactly how the data transfer is accomplished.

Fig. 7 shows a flowchart of the handshake sequence. When the first event, NRFD, goes false, this tells the talker that all of the listeners are now ready to receive a new data byte. The slowest listener is the last one to release NRFD, which will go high.

Next, the talker puts the data byte on the DIO lines and waits briefly to let the signals settle (usually about 10 μs). Once the data is on the DIO lines, NRFD is checked by the talker; if it is false, the talker sets DAV to true. The listeners now know that the new data is ready for pickup. (If NRFD is true, the talker waits until it goes false.)

The first listener that detects DAV true now sets NRFD true, and all of the listeners pick up the data byte from the DIO lines. Up to now, NDAC has been true, and as each listener gets its

Z8000.
You know it's better. Now, it's real.

Welcome to the next microcomputer revolution. A whole new generation of processors that bring big-machine architecture to small computers: Up to 10 times the throughput of Z80. *Eight megabytes* of directly-addressable memory. Plus, a rich, powerful set of over 400 instructions, data types and address modes.

Not a promise. A reality.

Introducing MPU-8000™— our brand-new 16-bit CPU that makes the most of Z8000's many advantages, and then some.

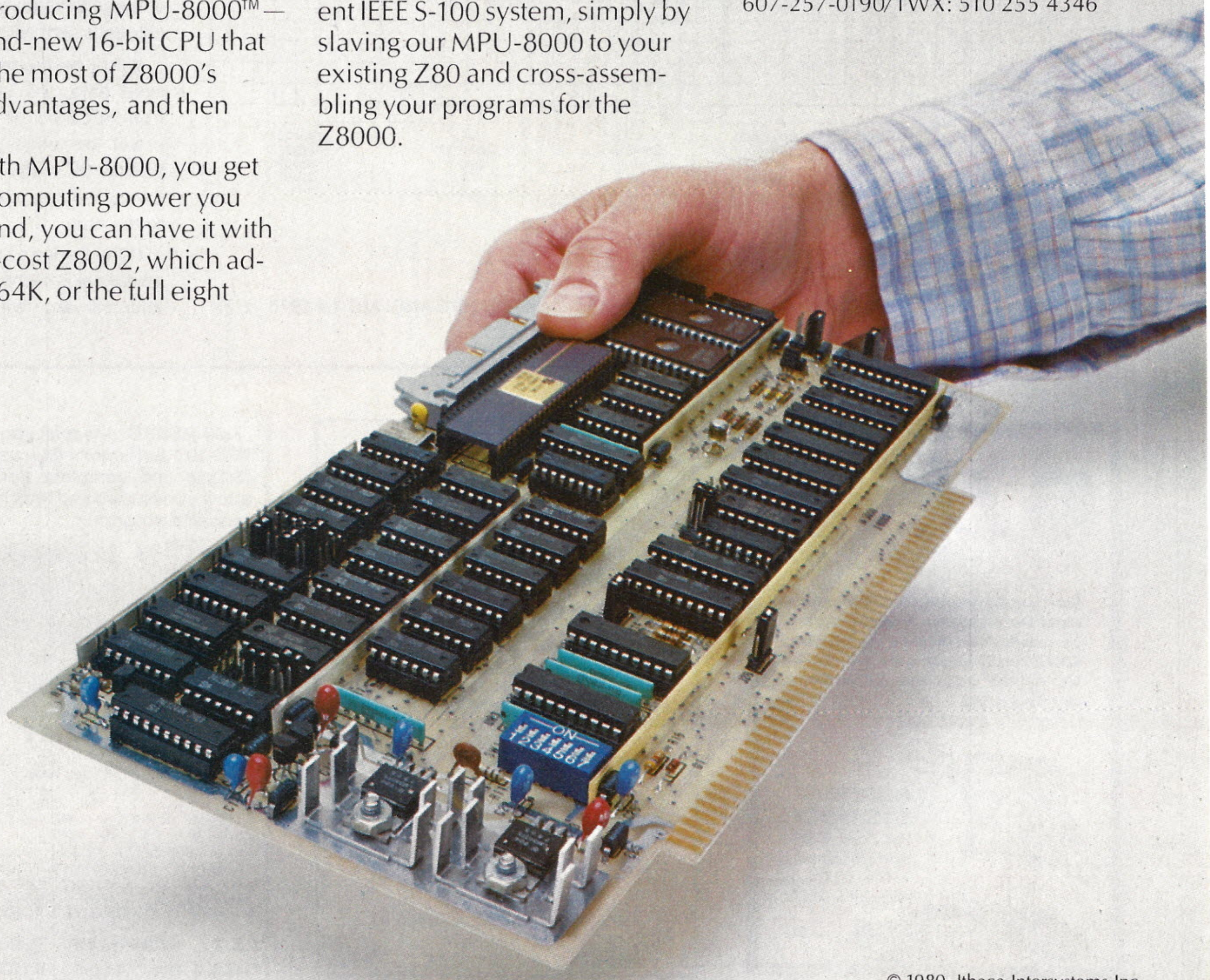
With MPU-8000, you get *all* the computing power you need. And, you can have it with the low-cost Z8002, which addresses 64K, or the full eight megabyte Z8001. There's plenty of room for expansion, too, via a top connector that makes adding an MMU or other Zbus-compatible accessories literally a snap.

Software support? Z8000 Pascal, editor, macro-assembler, operating system and more are on the way. But you don't even have to wait for them. Our new CPU is usable *now* in your present IEEE S-100 system, simply by slaving our MPU-8000 to your existing Z80 and cross-assembling your programs for the Z8000.

Think about it: 16-bit S-100. All the power, speed and flexibility of a mini, simply by plugging in the new MPU-8000. From Intersystems: Computer products you can rely on. And build on, too.

InterSystems™

Ithaca Intersystems Inc.,
1650 Hanshaw Road/P.O. Box 91,
Ithaca, NY 14850
607-257-0190/TWX: 510 255 4346



byte, it releases NDAC. NDAC goes false when all the listeners have the data. The talker waits for NDAC to go false, and when it does, the talker sets DAV to false. The listeners then make NDAC true, and the entire handshake sequence begins again.

Since a device is either a listener, talker or not addressed, Fig. 7 is broken into two flowcharts: one for the talker and one for the listener. A listener will start the handshake with NRFD and NDAC true, while the talker checks these. If both are false—the listener isn't there—an error condition exists.

Commands and Messages

When ATN is true, any data on DIO is seen as a command. Fig. 8 shows the entire ASCII set of 128 characters devoted to IEEE 488 commands.

Table 1. All PET I/O lines.

IEEE 488 PIA (6520)		ADDRESS: $ E820	59424
PAØ	IEEE Data In 1	PBØ	IEEE Data Out 1
PA1	" " 2	PB1	" " 2
PA2	" " 3	PB2	" " 3
PA3	" " 4	PB3	" " 4
PA4	" " 5	PB4	" " 5
PA5	" " 6	PB5	" " 6
PA6	" " 7	PB6	" " 7
PA7	" " 8	PB7	" " 8
CA1	ATN In	CB1	SRQ In
CA2	NDAC Out	CB2	DAV Out

MULTILINE INTERFACE MESSAGES: ISO-7 BIT CODE REPRESENTATION
(SENT AND RECEIVED WITH ATN=1)

BITS ② b_4	b_3	b_2	b_1	ROW \ COL. (b_7 b_6 b_5)	0 (0 0 0)	MSG ①	1 (0 0 1)	MSG	2 (0 1 0)	MSG	3 (0 1 1)	MSG	4 (1 0 0)	MSG	5 (1 0 1)	MSG	6 (1 1 0)	MSG	7 (1 1 1)	MSG
0	0	0	0	0	NUL		DLE		SP	MLA ASSIGNED TO DEVICE	0	MLA ASSIGNED TO DEVICE	●	MTA ASSIGNED TO DEVICE	P	MTA ASSIGNED TO DEVICE		MEANING DEFINED BY PCG CODE	p	MEANING DEFINED BY PCG CODE
0	0	0	1	1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
0	0	1	0	2	STX		DC2		"		2		B		R		b		r	
0	0	1	1	3	ETX		DC3		#		3		C		S		c		s	
0	1	0	0	4	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
0	1	0	1	5	ENQ	PPC③	NAK	PPU	%		5		E		U		e		u	
0	1	1	0	6	ACK		SYN		&		6		F		V		f		v	
0	1	1	1	7	BEL		ETB				7		G		W		g		w	
1	0	0	0	8	BS	GET	CAN	SPE	(		8		H		X		h		x	
1	0	0	1	9	HT	TCT	EM	SPD	)		9		I		Y		i		y	
1	0	1	0	10	LF		SUB		*		:		J		Z		j		z	
1	0	1	1	11	VT		ESC		+		;		K		[		k		{	
1	1	0	0	12	FF		FS		,		<		L		\		l		:	
1	1	0	1	13	CR		GS		-		=		M		]		m		}	
1	1	1	0	14	SO		RS		.		>		N		⌒		n		~	
1	1	1	1	15	SI		US		/	④	?	UNL	O		—	UNT	o		DEL	
					ADDRESSED COMMAND GROUP (ACG)		UNIVERSAL COMMAND GROUP (UCG)		LISTEN ADDRESS GROUP (LAG)				TALK ADDRESS GROUP (TAG)				SECONDARY COMMAND GROUP (SCG)			

PRIMARY COMMAND GROUP (PCG)

NOTES: ① MSG = INTERFACE MESSAGE
② b_1 = DIO1 ... b_7 = DIO7
③ REQUIRES SECONDARY COMMAND
④ DENSE SUBSET (COLUMN 2 THROUGH 5)

Fig. 8. IEEE 488 command set reproduced from the IEEE Standard 488-1975/ANSI MC 1.1-1975, *p. 77.*

The ASCII codes 32 through 62 (all numbers in decimal) designate the listen address for a device. Most IEEE-488-compatible devices have a five-position DIP switch next to the 488 connector set to the device's address, a number from 0 to 31. (Note: For the PET, use 4–15.) When the listen address is sent with ATN true and this address matches the device's address, the device will now be addressed to listen and will accept any data sent with ATN false.

If the device is supposed to send data, the talk address—from ASCII codes 64 through 94—will be used instead. The device (if with matching address) will now send data bytes to the bus.

If the device's address (by the switches) is number 7, the listen address value will be 32 + 7, or

KEYBOARD PIA (6520)	ADDRESS: $ E810 59408
PAØ Keyboard Row Select, LSB	PBØ Keyboard Column A
PA1 " " "	PB1 " " B
PA2 " " "	PB2 " " C
PA3 " " " , MSB	PB3 " " D
PA4 Switch, Cassette #1	PB4 " " E
PA5 " " #2	PB5 " " F
PA6 EOI In	PB6 " " G
PA7 Diagnostic Jumper	PB7 " " H
CA1 Read, Cassette #1	CB1 Video Horiz Sync In
CA2 Screen Blank & EOI Out	CB2 Motor, Cassette #1

The Diagnostic LED will light if PAØ-High, PA1-High, PA2-Low, PA3-High.

USER PORT VIA (6522)	ADDRESS: $ E840 59456
PAØ User Port LSB	PBØ NDAC In
PA1 " "	PB1 NRFD Out
PA2 " "	PB2 ATN Out
PA3 " "	PB3 Write, Both Cassettes
PA4 " "	PB4 Motor, Cassette # 2
PA5 " "	PB5 Video Horiz Sync In
PA6 " "	PB6 NRFD In
PA7 " " MSB	PB7 DAV In
CA1 User Port Handshake	CB1 Read, Cassette #2
CA2 Characters ROM Select	CB2 User Port Handshake

CA2 selects the MSB of the characters ROM, selecting the PET's graphics or lower case characters for the display.

39 (apostrophe). The talk address will be 64 + 7, or 71 (letter G). Notice that bits 5-7 designate talk or listen, and bits 0-5 designate the address. Address 31 is reserved for two special commands. Although you can set the switches on a device to 31, it won't operate with this setting.

One instrument must provide these talk and listen addresses. This device is the controller, and the PET is always the controller. The controller can talk and listen too, but only the controller can set ATN true.

Two of the ASCII codes, 63 and 95, serve as "universal" commands. The 63 code is known as "unlisten" and tells all addressed devices to stop listening to the bus. This is faster than trying to tell the devices one at a time to stop listening. The 95 code, "untalk," stops all data transmitters (talkers).

When a message—or a group of data bytes—is sent on the 488 bus, the controller sets ATN true and sends a listen address; the controller sets ATN true and sends a talk address; the talker puts data on the bus, and the listener picks it up. When the talker is finished, it may set EOI true on the last byte or send CR LF as the last bytes. The controller now sets ATN true and sends untalk (UNT) and unlisten (UNL), which reset the two devices.

In many cases, the controller—in this case, the PET—does the talking or listening. The controller can make everything stop by either setting IFC true or setting ATN true and putting UNT on the bus. Since UNT has its five lowest significant bits true, the active low operation of the IEEE lines overrides whatever data is present. (In normal operation of the bus, the controller doesn't have to take these

Listing 1. Memory Monitor.

```
10 PRINT"clr sp sp sp sp sp sp sp --> MEMORY MONITOR <--
20 PRINT"dn sp sp THIS PGM DISPLAYS A LOCATION IN THE
30 PRINT"PET'S MEMORY IN BOTH DECIMAL AND IN A
40 PRINT"'FRONT PANEL' FORMAT.
50 PRINT"dn sp sp YOU CAN CHANGE THE ADDRESS OR VALUE
60 PRINT"BY ENTERING A NEW VALUE WHEN THE '>>'
70 PRINT"MARKER IS NEXT TO THE ITEM YOU ARE
80 PRINT"CHANGING.
90 PRINT"dn sp sp PRESS 'RETURN' TO ENTER THE CHANGE
100 PRINT"OR TO MOVE THE MARKER.
110 PRINT"dn sp sp THE PGM CONSTANTLY PEEKS THE LOCATION
120 PRINT"WHEN YOU AREN'T CHANGING A VALUE. IF
130 PRINT"YOU CHANGE THE ADDRESS, THE PGM
140 PRINT"WILL SHOW THE NEW VALUE. IF YOU CHANGE
150 PRINT"A VALUE, IT IS POKED INTO MEMORY.
160 PRINT"dn sp sp 'H' WILL GIVE YOU SOME HELP FOR EACH
170 PRINT"ITEM.
190 PRINT"dn PRESS ANY KEY TO START
195 GETA$:IFA$=""THEN195

200 REM DRAW DISPLAY FORMAT
210 PRINT"clr dn dn dn dn dn dn";
220 D1$="@ @ @ @ @ @ @ @ @ @ @ @ @ @ @"      (15 shift-@)
230 D3$="rt rt rt rt rt rt rt rt rt rt"      (10 rt's)
240 PRINT"rt rt rt rt rt rt rt -->sp MEMORY MONITOR sp<--"
250 PRINTD3$"Ø"D1$"."
255 PRINTD3$"] ADDRESS:sp sp sp sp sp sp sp sp ]"   (] is a vertical line)
260 PRINTD3$"+"D1$"3"
265 PRINTD3$"] 7 sp 6 sp 5 sp 4 sp 3 sp 2 sp 1 sp Ø ]"
270 PRINTD3$"+ @ 2 @ 2 @ 2 @ 2 @ 2 @ 2 @ 3"
280 PRINTD3$"] sp ] sp ] sp ] sp ] sp ] sp ] sp ] sp ]"
290 PRINTD3$"+ @ 2 @ 2 @ 2 @ 2 @ 2 @ 2 @ 3"
300 PRINTD3$"] DECIMAL: sp sp sp sp sp sp sp ]"
310 PRINTD3$"-"D1$"="
320 PRINT"dn dn dn"
```

NOTE: For Lines 200-320 see Fig. 9.

```
400 REM IDLING PROGRAM
410 AD=59471:PT=1
500 REM DISPLAY ADDRESS
510 GOSUB 1000
520 REM DISP PANEL LITES
525 DT=PEEK(AD)
530 GOSUB 1200
540 REM DISP DECIMAL
550 GOSUB 1400
560 REM DISP PTR
570 GOSUB 1600
580 REM GET CHAR
590 GET A$
600 IF A$="" THEN 500
610 IF A$=CHR$(13) THEN 700
620 FG=Ø:GOSUB 2500
630 IF FG=Ø THEN 510
640 GOTO 210
700 REM BUMP PTR
710 GOSUB 1800
720 GOTO 510

1000 REM DISP ADDR
1010 PRINT"hm dn dn dn dn dn dn dn"
1020 V$=STR$(AD)+"sp sp sp sp sp sp sp"
1030 V$=MID$(V$,2,6)
1040 PRINT TAB(2Ø);V$
1050 RETURN

1200 REM DISP PANEL
1210 PRINT"dn dn dn"TAB(11);
1220 VT=DT:DV=128
1230 FOR J=1 TO 8
1240 IF VT/DV< 1 THEN 1260
1250 PRINT" Q rt";:VT=VT-DV:GOTO 1300
1260 PRINT" W rt";
1300 DV=DV/2
1310 NEXT J

1400 REM DISP DECIMAL
1410 PRINT"dn"
1420 V$=STR$(DT)+" sp sp sp sp sp sp sp sp "
1430 V$=MID$(V4,2,6)
1440 PRINT TAB(2Ø);V$
1450 RETURN

1600 REM DISP PTR
1610 PRINT"hm dn dn dn dn dn dn dn"
1620 IF PT >1 THEN 1640
1630 PRINT TAB(8)">>";:RETURN
1640 PRINT"dn dn dn"
1650 IF PT >2 THEN 1670
1660 GOTO 1630
1670 PRINT"dn":GOTO 1630

1800 REM BUMP PTR
1810 GOSUB 1600
1820 PRINT"rt rt sp sp"
1830 PT=PT+1: IF PT >3 THEN PT=1
1840 GOSUB 1600
1850 RETURN

2000 RETURN            (This line probably isn't needed)

2500 REM CHANGE ITEM
2510 ON PT GOSUB 3000,3500,4000
2520 RETURN

3000 REM CHANGE ADDR
3010 IF A$="H" THEN GOSUB 4500:RETURN
```

VALUES FOR INPUTS

IEEE LINE	ADDRESS (HEX)	ADDRESS (DECIMAL)	BIT
DIO 1	E820	59424	0
DIO 2	E820	59424	1
DIO 3	E820	59424	2
DIO 4	E820	59424	3
DIO 5	E820	59424	4
DIO 6	E820	59424	5
DIO 7	E820	59424	6
DIO 8	E820	59424	7
EOI	E810	59408	6
IFC	----	-----	-
SRQ	E823	59427	7
REN	----	-----	-
ATN	E821	59425	7
DAV	E840	59456	7
NRFD	E840	59456	6
NDAC	E840	59456	0

VALUES FOR OUTPUTS

IEEE LINE	ADDRESS (HEX)	ADDRESS (DECIMAL)	BIT
DIO 1	E822	59426	0
DIO 2	E822	59426	1
DIO 3	E822	59426	2
DIO 4	E822	59426	3
DIO 5	E822	59426	4
DIO 6	E822	59426	5
DIO 7	E822	59426	6
DIO 8	E822	59426	7
EOI	E811	59409	3
IFC	----	-----	-
SRQ	----	-----	-
REN	----	-----	-
ATN	E840	59456	2
DAV	E823	59427	3
NRFD	E840	59456	1
NDAC	E821	59425	3

Table. 2 Addresses and bits for the IEEE 488 lines.

drastic measures.)

In some cases, a device will have a secondary address, which permits more than 31 effective addresses on the bus. For example, the Commodore printer might be set as device 4. To control internal functions, secondary addresses select the function in use. (See Commodore's "PET Communication with the Outside World," p. 19.) If a secondary address is in use, it is sent immediately after the talk or listen address, known as the primary address, with ATN true.

Several of the bus-management lines, such as SRQ, EOI, REN and IFC, serve special functions. Many instruments do respond to these, and often the response depends upon the instrument.

When ATN is low, about half the ASCII code is devoted to special commands, which come in defined sequences whose definition takes about two-thirds of the formal IEEE 488 specification. Most instruments use only a few of these.

Flipping Bits

The PET ultimately communicates to the rest of the world by the screen and some interface chips—two 6520s and one 6522. (For the specs on these chips, contact MOS Technology.) The 6520 and 6522 chips can only drive one TTL load, so the PET's IEEE lines are connected to some buffer chips to provide the currents needed in the IEEE 488 bus.

Table 1 indicates all of the PET's I/O line assignments as a reference. The PET utilizes all 60 I/O lines as shown here. Most of the IEEE lines are buffered with MC 3446P bidirectional line

```
3020 V1=AD
3030 GOSUB 5000
3040 IF V2<Ø THEN RETURN
3050 IF V2>65535 THEN RETURN
3060 AD=V2:RETURN

3500 REM CHANGE BINARY VALUE
3510 IF A$="H"THEN GOSUB 4600: RETURN
3520 V1=DT
3530 GOSUB 5500
3540 IF V2<Ø THEN RETURN
3550 IF V2>255 THEN RETURN
3560 DT=V2:POKE AD,DT:RETURN

4000 REM CHANGE VALUE
4010 IF A$="H" THEN GOSUB 4500: RETURN
4020 V1=DT
4030 GOSUB 5000
4040 IF V2<Ø THEN RETURN
4050 IF V2>255 THEN RETURN
4060 DT=V2:POKE AD,DT: RETURN

4500 PRINT"clr sp sp TYPE IN THE NEW NUMBER AND PRESS
4505 FG=1
4510 PRINT"RETURN. PRESS 'X' TO ABORT & NOT MAKE
4515 PRINT"THE CHANGE.
4520 PRINT" sp sp PRESS SPACE TO ERASE REST OF NUMBER.
4530 PRINT"dn sp sp PRESS ANY KEY
4540 GETA$: IFA$=""THEN 4540
4550 RETURN

4600 PRINT"clr sp sp ENTER '1' OR ' Q ' TO SET A BIT, AND
4610 PRINT"'Ø' OR ' W ' TO RESET A BIT. PRESS
4620 PRINT"RETURN WHEN DOEN.
4625 PRINT" sp sp PRESS SPACE TO SKIP A BIT.
4630 PRINT"dn sp sp PRESS ANY KEY
4640 GETA$:IF A$="" THEN 4540
4650 RETURN

5000 REM NUMERIC ENTRY
5010 REM POS CURSOR
5020 PRINT TAB(20);
5030 REM MAKE DISP STR
5040 D$=MID$(STR$(V1),2)+"sp sp sp sp sp sp sp"
5050 D$=LEFT$(D$,6)
5060 REM SET RVS PTR & RETURN VALUE
5070 PC=1:V2=-1
5080 REM SEE INPUT & ACT
5090 IF A$="X" THEN RETURN
5100 IF A$=CHR$(13) THEN V2=VAL(D$):RETURN
5110 IF A$<>" sp " THEN 5120
5112 IF PC=1 THEN D$="sp sp sp sp sp sp":GOTO 5210
5114 D$=LEFT$(D4,PC-1)+"sp sp sp sp sp sp":D$=LEFT$(D$,6)
5118 GOTO 5210
5120 IF A$<"Ø" OR A$>"9" THEN 5210
5125 REM REMAKE STRING
5130 DX$=D$:D$=""
5140 FOR J=1 TO 6
5150 IF PC=J THEN D$=D$+A$:GOTO 5170
5160 D$=D$+MID$(DX$,J,1)
5170 NEXT J
5180 PC=PC+1: IF PC>7 THEN PC=1

5200 REM DISPLAY RESULT & RESTORE CURSOR
5210 FOR J=1 TO 6
5220 IF J=PC THEN PRINT "rvs";
5230 PRINT MID$(D$,J,1);
5240 IF J=PC THEN PRINT "off";
5250 NEXT J:PRINT"lft lft lft lft lft lft";
5260 GET A$: IF A$="" THEN 5260
5270 GOTO 5090
5500 REM BINARY ENTRY
5510 PRINT TAB(11);
5520 FOR J= 1 TO 8
5525 V1=V1/2
5530 IF V1=INT(V1) THEN D$=" W "+D$: GOTO 5540
5535 D$=" Q "+D$
5540 V1 = INT(V1)
5550 NEXT J
5570 REM SET RVS PTR
5580 PC=1:V2=-1
5590 REM LOOK AT INPUT
5600 IF A$="X" THEN RETURN
5605 IF A$=CHR$(13)THEN 5780
5610 IF A$="sp" THEN 5715
5620 IF A$="1" OR A$=" Q " THEN A$=" Q ":GOTO 5660
5630 IF A$="Ø" OR A$=" W " THEN A$+" W ":GOTO 5660
5640 GETA$:IFA$=""THEN 5640
5650 GOTO 5600
5660 REM REMAKE STRING
5670 DX$=D$:D$=""
5680 FOR J= 1 TO 8
5690 IF PC=J THEN D$=D$+A$: GOTO 5710
5700 D$=D$+MID$(DX$,J,1)
5710 NEXT J
5715 PC=PC+1:IF PC>8 THEN PC=1
5720 REM DISP & FIX CURSOR
5730 FOR J= 1 TO 8
5735 IF J = PC THEN PRINT "rvs";
5740 PRINT MID$(D$,J,1)"rt";
5745 IF J=PC THEN PRINT "off";
5750 NEXT J:PRINT" lft lft lft lft lft lft lft lft
                   lft lft lft lft lft lft lft lft";  (16 lft's)
5760 GOTO 5640
5770 REM MAKE VALUE
5780 V2=Ø:FORJ=1 TO 8
5785 V2=V2*2
5790 IF MID$(D$,J,1)=" W " THEN 5810
5800 V2=V2+1
5810 NEXT J
5820 RETURN
```

driver chips to provide the IEEE current requirements. SRQ is an input only and connects directly to the 6520 chip. IFC is buffered with a NAND and some resistors to the IEEE specification.

All DIO Lines:*	
IN:	POKE 59426,255:V = PEEK(59424):V = (NOT(V))AND255
OUT:	V = (NOT(V))AND255: POKE 59426,V
EOI	
IN:	V = 1:IF PEEK(59408)AND 64 THEN V = 0
TRUE OUT:	POKE 59409, PEEK(59409) AND 247
FALSE OUT:	POKE 59409, PEEK(59409) OR 8
REN & IFC—Not Applicable	
SRQ**	
IN:	V = 0:IF PEEK(59427) AND 128 THEN V = 1
	Z = PEEK(59426)
LO-HI	POKE 59427, PEEK(59427) OR 2
HI-LO	POKE 59427, PEEK(59427) AND 253
ATN**	
IN:	V = 0:IF PEEK(59425) AND 64 THEN V = 1
	Z = PEEK(59424)
LO-HI:	POKE 59409, PEEK(59409) OR 2
HI-LO:	POKE 59409, PEEK(59409) AND 253
TRUE OUT:	POKE 59456, PEEK(59456) AND 251
FALSE OUT:	POKE 59456, PEEK(59456) OR 4
DAV	
IN:	V = 1: IF PEEK(59456) AND 128 THEN V = 0
TRUE OUT:	POKE 59427, PEEK(59427) AND 247
FALSE OUT:	POKE 59427, PEEK(59427) OR 8
NRFD	
IN:	V = 1: IF PEEK(59456) AND 64 THEN V = 0
TRUE OUT:	POKE 59456, PEEK(59456) AND 253
FALSE OUT:	POKE 59456, PEEK(59456) OR 2
NDAC	
IN:	V = 1: IF PEEK(59456) AND 1 THEN V = 0
TRUE OUT:	POKE 59425, PEEK(59425) AND 247
FALSE OUT:	POKE 59425, PEEK(59425) OR 8

*The extra parenthesis in the complementation of V is required, for the PET evaluates AND before NOT.

**The HI-LO or LO-HI determines which transition the CA/CB1 inputs will respond to. Set the HI-LO or LO-HI before doing the IN: line. The Z = PEEK resets the flag bit. Be sure to reset the flag bit before checking the first time.

SRQ OUT is not available on the PET.

Table 3. PEEKs and POKEs for the IEEE 488 lines.

Table 1 reveals some interesting irregularities concerning the IEEE 488 bus: If EOI is true, the PET's display is turned off. (Programs that PEEK and POKE the display area in memory can use this to avoid snow.) Later-model PETs don't have this problem. REN isn't listed; the PET's REN line is wired to ground (true). IFC is not shown. The PET's IFC is connected to the power-on one-shot, which sets IFC true for about 100 ms when the PET is turned on. If you reset the PET by grounding the $\overline{\text{RES}}$ line, IFC may not go true. A better approach is to trigger the power-on one-shot by inserting a switch between power and the 555's power pin. The SRQ line is an input only. The PET's firmware does not use SRQ, so you have to program it directly.

In a 650x-based system, all I/O is seen as a set of memory addresses. This means that BASIC's PEEK and POKE can be used to control the IEEE 488 lines. Table 2 indicates the addresses and bits involved for the PET's IEEE lines. In most cases, a direct PEEK or POKE will do. Two lines, ATN in and SRQ in, require a more complex sequence. These are connected to CA1 and CB1 of a 6520, which set flag bits in the Interrupt Flag register. Resetting these bits requires a memory access to the DIO data register.

Table 3 lists the specific PEEKs and POKEs to individually sense or modify the IEEE lines. In many cases the PEEK or POKE values can be ANDed or ORed together to do several operations at once. If you have built the IEEE Blinkin Lites, try a few of these PEEKs and POKEs to see how they work.

When I was flipping bits with PEEK and POKE for the IEEE lines, I was confused each time I had to figure out the decimal numbers for each changed bit. Perhaps it would be easier to display a byte of memory on the PET's screen in a "front panel" format with simulated LEDs for each bit and some simple keyboard commands to change bits and addresses. Memory Monitor (see Listing 1) does this.

When Memory Monitor is loaded and run, and the first page of instructions is read, the display in Fig. 9 is shown. A box with four parts appears in the middle of the screen with the title Memory Monitor placed above the box. Left of the box is a marker, >>, which indicates the part of the box accessible by the keyboard.

The top of the box shows the

Listing 2. BASIC 488 program.

```
1000 REM **** IEEE 488 ****
1005 REM   GREGORY YOB, JAN 1979
1010 REM   BOX 354, PALO ALTO CA 94301
1015 REM
1020 REM THESE ROUTINES PERMIT DIRECT
1025 REM MANIPULATION OF THE PET IEEE
1030 REM 488 BUSS LINES AND (SLOW!)
1035 REM IEEE 488 COMMAND AND DATA
1040 REM TRANSFERS
1045 REM
1500 REM -- INITIALIZATION --
1510 RESTOR:READ A1,A2,A3,A4,A5,A6,A7
1520 DATA 59424,59426,59425,59427,59408,59456,59409
1530 READ MØ,M1,M2,M3,M4,M5,M6,M7
1540 DATA 1,2,4,8,16,32,64,128
1550 READ NØ,N1,N2,N3,N4,N5,N6,N7
1560 DATA 254,253,251,247,239,223,191,127
1570 READ N8
1580 DATA 255
1590 READ O1,O2,O3,O4,O5,O6,O7     (Each of these is Letter O)
1600 DATA 255,266,60,60,249,255,60
1610 DEF FNF(X)=(NOT(X))AND255
1620 RETURN

7000 PRINT"clr GET MESSAGE"
7010 PRINT"dn PRESS KEY TO START"
7020 GETA$:IFA$=""THEN RETURN       ("" is an empty string)
7030 D2=FNF(DV+64):GOSUB9450:GOSUB8500:GOSUB9470
7040 B$=""
7050 GOSUB 8000:IF FNF(D1)=13THEN7070
7060 B$=B$+CHR$(FNF(D1)):GOTO7050
7070 GOSUB8000:REM LF BUCKET
7080 PRINT"dn dn MESSAGE IS: sp"B$
7090 RETURN

7500 PRINT"clr SEND MESSAGE"
7510 INPUT"dn dn MESSAGE:";C$
7520 D2=FNF(DV+32):GOSUB9450:GOSUB8500:GOSUB9470
7530 FOR J=1 TO LEN(C$)
7540 D2=FNF(ASC(MID$(C$,J)))
7550 GOSUB8500:NEXTJ
7560 PRINT"dn dn MESSAGE SENT: sp"C$
7570 RETURN

8000 PRINT"clr LISTEN HANDSHAKE dn"
8010 GOSUB9350:GOSUB9250:GOSUB9370
8020 PRINT" sp NRFD TRUE dn":PRINT" sp NDAC TRUE"
     :PRINT" sp NRFD FALSE"
8030 PRINT"WAITING FOR DAV TRUE"
8040 GETA$:IFA$<>""THENPRINT"--FORCED":GOTO8060
8050 GOSUB9100:IFH1=ØTHEN8040
8060 GOSUB9000:PRINT"dn spDATA:"FNF(D1)CHR$(FNF(D1))
8070 GOSUB9350:GOSUB9270
8080 PRINT"dn sp NRFD TRUE":PRINT" sp NDAC FALSE"
8090 PRINT"WAITING FOR DAV FALSE"
8100 GETA$:IFA$<> ""THENPRINT"--FORCED":GOTO8120
8110 GOSUB9100:IFH1=1THEN8100
8120 GOSUB9250
8130 PRINT"dn sp NDAC TRUE"
8140 RETURN

8500 PRINT"clr TALK HANDSHAKE"
8510 GOSUB9170
8520 PRINT"dn sp DAV FALSE"
8530 GOSUB9200:GOSUB9300
8540 IF H1+H2>Ø THEN 8570
8550 PRINT"dn>> sp ERROR STATE-PRESS KEY TO FORCE"
8555 PRINT"NOTE: MAKE NRFD, NDAC TRUE"
8560 GETA$;IFA$=""THEN8560
8570 GOSUB9050
8580 PRINT"dn DATA ON LINE:"FNF(D2)CHR$(FNF(D2))
```

address of a memory location in decimal. If you press SPACE, the address will be erased, and a new number can be entered. Pressing number keys enters a new address, and a reverse-field cursor appears.

When a cursor isn't on the screen, pressing RETURN will move the marker to the next part of the box. (The second part in the box indicates the bit numbers and is skipped by the marker.)

The third part of the box displays a front panel made of solid or hollow "balls" (or "LEDs"). This shows the eight bits of the byte under investigation. The numbers above the "LEDs" indicate the bit numbers, 7 the MSB and 0 the LSB. To change the byte, enter 0 or 1 (or Shift-Q and Shift-W), and the cursor will appear. Pressing RETURN enters the value.

The fourth part of the box is the value of the byte in decimal and is entered in the same way as the address.

If you press RETURN several times, the marker rotates through the three accessible parts of the box. To recall how to enter a value, press the letter H, which clears the screen and provides instructions.

The Memory Monitor eased the tedium and frustration of checking the PEEKs and POKEs used in the IEEE 488 memory locations. I have made Memory Monitor simple to use, and I consider it a good example of user-oriented programming.

Doing It the Hard Way

With direct access to the PET's IEEE 488 lines, you can use PEEK and POKE to operate an IEEE instrument "by hand." This is probably more difficult than using the IEEE Blinkin Lites box to communicate switch by switch because it takes more keystrokes to change a bit with POKE.

The next step is to write a BASIC program that performs the required IEEE 488 operations directly. Though the PET has these "built in," there are a few advantages to doing the whole thing in BASIC.

Everything goes slowly. As events happen, there is a chance of seeing them as they go by.

BASIC is accessible. If the PET or your instrument decides that the sky's the limit, pressing the STOP key can illuminate where the difficulties lie. The PET's built-in IEEE 488 services are mostly invisible, and there's often no way to find out why something went wrong.

Everything is under control. It is simple enough to display every step with suitable messages to the screen. If necessary, you can insert a GET loop to make the PET wait until a key is pressed before proceeding.

Changes are easy.

It's an educational experience—those who must learn the "nuts and bolts" of the IEEE bus will find a BASIC emulator useful.

I constructed the BASIC 488 program (see Listing 2) to provide the following essential services: put the PEEK and POKE values into variable form for reasonably fast execution and to simplify debugging with direct

```
-->  MEMORY MONITOR       <--
>>  | ADDRESS: 59471     |
    | 7  6  5  4  3  2  1  0 |
    | •  •  •  •  •  •  •  • |
    | DECIMAL: 255       |
```

Fig. 9. Listing 1's initial display.

```
8590 print"dn WAITING FOR NRFD FALSE"
8600 GETA$:IFA$<>""THENPRINT"--FORCED":GOTO8620
8610 GOSUB9300:IFH3=1THEN8600
8620 GOSUB9150
8630 PRINT"dn sp DAV TRUE"
8640 PRINT"WAITING FOR NDAC FALSE"
8650 GETA$:IFA$<>""THEN8670
8670 GOSUB9170
8680 PRINT"dn sp DAV FALSE"
8690 RETURN

9000 POKEA2,N8:D1=PEEK(A1):RETURN
9050 POKEA2,D2:RETURN
9100 H1=1:IFPEEK(A6)ANDM7THENH1=Ø
9110 RETURN
9150 POKEA4,PEEK(A4)ANDN3:RETURN
9170 POKEA4,PEEK(A4)ORM3:RETURN
9200 H2=1:IFPEEK(A6)ANDMØTHENH2=Ø
9210 RETURN
9250 POKEA3,PEEK(A3)ANDN3:RETURN
9270 POKEA3,PEEK(A3)ORM3:RETURN
9300 H3=1:IFPEEK(A6)ANDM6THENH3=Ø
9310 RETURN
9350 POKEA6,PEEK(A6)ANDN1:RETURN
9370 POKEA6,PEEK(A6)ORM1:RETURN
9400 PRINT"NO ATN LEVEL":STOP
9430 H4=Ø:IFPEEK(A3)ANDM7THENH4=1
9440 ZZ=PEEK(A1):RETURN
9450 POKEA6,PEEK(A6)ANDN2:RETURN
9470 POKEA6,PEEK(A6)ORM2:RETURN
9500 H5=1:IFPEEK(A5)ANDM6THENH5=Ø
9510 RETURN
9550 POKEA7,PEEK(A7)ANDN3:RETURN
9570 POKEA7,PEEK(A7)ORM3:RETURN
9600 REM SRQ NOT OUTPUT
9630 H6=Ø:IFPEEK(A4)ANDM7THENH6=1
9640 ZZ=PEEK(A2):RETURN
```

Entry Points:

SUBROUTINE 1500	Initialization (Must be done first)
SUBROUTINE 7000	Get Message as B$, Requires DV
SUBROUTINE 7500	Put Message C$, Requires DV
SUBROUTINE 8000	Listen Handshake
SUBROUTINE 8500	Talk Handshake
SUBROUTINES 9000ffl9600	IEEE Lines Primitives
9000	Read DIO as D1
9050	Write DIO as D2
9100	Read DAV as H1
9150	Set DAV TRUE
9170	Set DAV FALSE
9200	Read NDAC as H2
9250	Set NDAC TRUE
9270	Set NDAC FALSE
9300	Read NRFD as H3
9350	Set NRFD TRUE
9370	Set NRFD FALSE
9400	Trap for ATN
9430	Check ATN as H4 (If changed)
9450	Set ATN TRUE
9470	Set ATN FALSE
9500	Read EOI as H5
9550	Set EOI TRUE (Screen will blank)
9570	Set EOI FALSE (Screen returns)
9630	Check SRQ as H6 (If changed)

Variables:

PEEK/POKE ADDRESSES		ORIGINAL VALUES	
A1	59424	01	255
A2	59426	02	255
A3	59425	03	60
A4	59427	04	60
A5	59408	05	249
A6	59456	06	255
A7	59409	07	60

Masks:

M0	0000 0001	1	N0	1111 1110	254
M1	0000 0010	2	N1	1111 1101	253
M2	0000 0100	4	N2	1111 1011	251
M3	0000 1000	8	N3	1111 0111	247
M4	0001 0000	16	N4	1110 1111	239
M5	0010 0000	32	N5	1101 1111	223
M6	0100 0000	64	N6	1011 1111	191
M7	1000 0000	128	N7	0111 1111	127
			N8	1111 1111	255

Miscellaneous:

DV	Device Address
A$	Keyboard dummy entry
B$	Message from Device
C$	Message to Device

Functions:

FNF(X)	Returns complement of argument

Table 4. BASIC 488 program notes.

```
CLR
GOSUB 1500:GOSUB 1900        Get everything ready . . .
PRINT FNF(32 + 7)
 216                         This is the value for D2 as a listen address.
GOSUB 9450                   Make ATN true.
D2 = 216:GOSUB 8500          Send listen address via handshake
TALK HANDSHAKE               The PET responds with the step-by-step
  DAV FALSE                  output handshake and goes successfully
DATA ON LINE: 39'            through the entire process.
WAITING FOR NRFD FALSE
  DAV TRUE
WAITING FOR NDAC FALSE       The HP Clock's "addressed" light turns on!
  DAV FALSE
READY.
GOSUB 9470                   Make ATN false
PRINT FNF(ASC("R"))          R resets the clock
 173
D2 = 173:GOSUB 8500          Send 'R' as data . . . . .
( . . . . . . . . . . )      And this handshakes through OK too.
```

Example 1. My dialogue with the HP clock via BASIC 488.

```
D2 = 216:GOSUB9450:GOSUB8500:GOSUB9470:D2 = 173:GOSUB8500
```

Example 2. A one-line command for Example 1.

```
PRINT FNF(64 + 7)                              Find out D2 for talk address
 184
D2 = 184:GOSUB9450:GOSUB8500:GOSUB9470
( . . . . . . . . . . . )                      The handshake goes through
FOR J = 1 TO 14:GOSUB 8000:NEXT
( . . . . . . for 14 times . . . . . )
```

Example 3. The dialogue for reading the clock.

commands; do most of the PEEKs and POKEs for line control as short subroutines; provide the listen and talk handshake sequences for one byte and display their progress; provide a way to send and receive strings to a device on the bus; set the program up as a skeleton onto which you can add specific programs to suit changing needs.

Table 4 indicates the subroutines and variables used in the BASIC 488 program. Load these subroutines and then add the code you need for your devices. Some devices, such as those by Commodore, may not follow the IEEE time standard, and the BASIC 488 program will not be fast enough to prevent time-outs.

I built the program from the bottom up, starting with subroutines 1500 and the series starting at 9000. Subroutine 1500 sets up the essential variables. A1-7 are the addresses of the PEEK/POKE locations; M0–M7 and N0–N8 are AND and OR masks to extract bits 0–7 from a location (or to set the desired bits); 01–07 are the original values for addresses A1–A7. (POKE A1,01, for example, will restore location A1 to the PET's power-on value, which helps you to recover from disasters.)

The variables H1 to H6 are the sense values for the IEEE lines. For example, if H1 is 1, the DAV line is true. If H1 is zero, DAV is false.

When you enter BASIC 488, enter lines 1000–1620 and lines 9000–9640 first. Use the IEEE Blinkin Lites to check that the subroutines in the 9000 series function correctly. First, GOSUB 1000 in direct mode to set things up. Then, GOSUB to the section under test and look at the Blinkin Lites to see what happened. A PRINT H1 will inform you of the sensing subroutines' results. Be sure to thoroughly test the 9000 series first! Nothing else will work if these don't!

If all else fails, refer to Tables

```
X$ = "":FORJ = 1TO14:GOSUB8000:X$ = X$ + CHR$(FNF(D1)):NEXT:PRINTX$
0101000520
```

Example 4. Putting the clock's message into X$, and the contents of X$.

1, 2 and 3 and try a few direct PEEKs and POKEs to ensure that the IEEE lines are functional.

Add lines 8000–8140 and lines 8500–8690, which you can check by attaching the 488 Blinkin Lites and carefully tracing through the handshake flowchart in Fig. 7. Again, it is essential to be sure these routines work correctly. An additional benefit is that you will learn the handshake sequence in detail.

Note that the data transferred, D1 or D2, must be complemented with the FNF function as it enters or leaves the IEEE bus. In some of the waiting loops, such as lines 8030–8050, a GET A$ check is inserted. If the instrument hangs up, pressing a key will force the handshake to proceed, and a suitable message will appear on the screen. As the handshakes proceed, their progress is reported to the screen for your reference.

Next, add lines 7000–7570. These routines require a device address, DV, to function correctly. Subroutine 7000 will fetch a message from a device, and subroutine 7500 will send a message. The strings B$ and C$ are used to store the messages.

Most devices will send an EOI along with the last character of their messages. This will turn off the screen. In some cases, you will have to provide an EOI, which will again turn off the screen. To recover, enter:

```
GOSUB 9570 (and RETURN)
```

Another approach is to move the cursor down until the screen scrolls. A scroll turns the screen off, and then on. If you have a 16K PET, the screen will not blink.

Testing the last part via the IEEE Blinkin Lites is tedious. If you have an instrument available, try talking to it! Be sure you know *exactly* what your instrument expects and its responses!

Talking to the HP Clock via BASIC 488

Now that you have checked out BASIC 488 by hand, try it with a real live instrument! I connected the HP clock, loaded BASIC 488 and gave it a try (see Example 1). The clock's front panel shows the reset worked.

These commands can be compressed to one line (see Example 2).

```
DV = 7:GOSUB7500
SEND MESSAGE

MESSAGE:? R                R for reset

GET MESSAGE

PRESS KEY TO START

( . . . . . A lot of Listen Handshakes . . . . )

MESSAGE IS:  0101000158
```

Example 5. Resetting the clock.

Next, try to read the clock. Address the clock to talk, then read the 14-character message shown in Example 3. If you look at the line DATA: on the display for the Listen Handshake, you can barely see the clock's message. A different version (see Example 4) will pick up the message and leave it later. Below the Listen Handshake display appears the clock's message: 0101000520

The BASIC 488 program has two routines for sending and reading entire strings via the IEEE 488. Subroutine 7000 addresses device DV to talk and

```
IEEE Bus Handshake Routine
- Main Program

1800 A200   LDX #00     prepare index register
1802 A9FB   LDA #FB     set ATN low
1804 2D40E8 AND E840
1807 8D40E8 STA E840
180A A928   LDA #28     MLA (28 for this device)
180C 8501   STA 01
180E 208018 JSR 1880    handshake into bus
1811 A908   LDA #08     GET
1813 8501   STA 01
1815 208018 JSR 1880    handshake
1818 A948   LDA #48     MTA
181A 8501   STA 01
181C 208018 JSR 1880    handshake
181F A9FD   LDA #FD     set NRFD low
1821 2D40E8 AND E840       (ready to receive data)
1824 8D40E8 STA E840
1827 A9F7   LDA #F7     and NDAC low also
1829 2D21E8 AND E821
182C 8D21E8 STA E821
182F A904   LDA #04     set ATN high
1831 0D40E8 ORA E840
1834 8D40E8 STA E840
1837 A008   LDY #08     ready to count 8 bytes
1839 20B018 JSR 18B0    handshake data from bus
183C A502   LDA 02      result to A
183E 9D0119 STA 1901,X  store in  1901+X
1841 E8     INX
1842 88     DEY
1843 D0F4   BNE 1839    jump if Y not zero
1845 A9FB   LDA #FB     set ATN low
1847 2D40E8 AND E840
184A 8D40E8 STA E840
184D A902   LDA #02     set NRFD high
184F 0D40E8 ORA E840
1852 8D40E8 STA E840
1855 A908   LDA #08     set NDAC high
1857 0D21E8 ORA E821
185A 8D21E8 STA E821
185D A95F   LDA #5F     UNT
185F 8501   STA 01
1861 208018 JSR 1880    handshake to bus
1864 A904   LDA #04     set ATN high
1866 0D40E8 ORA E840
1869 8D40E8 STA E840
186C CE0019 DEC 1900    decrease counter
186F D091   BNE 1802    jump if not zero
1871 60     RTS         return to BASIC program

Subroutine to Handle
Handshake Into Bus

1880 AD40E8 LDA E840    NRFD ?
1883 2940   AND #40
1885 F0F9   BEQ 1880    jump back if not ready
1887 A501   LDA 01      ready:  get data byte
1889 49FF   EOR #FF     complement it
188B 8D22E8 STA E822    send to bus
188E A9F7   LDA #F7     set DAV low
1890 2D23E8 AND E823
1893 8D23E8 STA E823
1896 AD40E8 LDA E840    NDAC ?
1899 2901   AND #01
189B F0F9   BEQ 1896    jump back if not accepted
189D A908   LDA #08     accepted;  set DAV high
189F 0D23E8 ORA E823
18A2 8D23E8 STA E823
18A5 A9FF   LDA #FF     255_10 into bus
18A7 8D22E8 STA E822
18AA 60     RTS         return to main

Subroutine to Handle
Handshake From Bus

18B0 A902   LDA #02     set NRFD high
18B2 0D40E8 ORA E840
1885 8D40E8 STA E840
18B8 AD40E8 LDA E840    DAV ?
18BB 2980   AND #80
18BD D0F9   BNE 18B8    jump back if not valid
18BF AD20E8 LDA E820    get data byte from bus
18C2 49FF   EOR #FF     complement
18C4 8502   STA 02      store in $ 0002
18C6 A9FD   LDA #FD     set NRFD low
18C8 2D40E8 AND E840
18CB 8D40E8 STA E840
18CE A908   LDA #08     set NDAC high
18D0 0D21E8 ORA E821
18D3 8D21E8 STA E821
18D6 AD40E8 LDA E840    DAV high ?
18D9 2980   AND #80
18DB F0F9   BEQ 18D6    jump back if not
18DD A9F7   LDA #F7     set NDAC low
18DF 2D21E8 AND E821
18E2 8D21E8 STA E821
18E5 A9FF   LDA #FF     255_10 into bus
18E7 8D22E8 STA E822
18EA 60     RTS         return to main

IEEE Bus Handshake Routine
Object Listing

1800 A2 00 A9 FB 2D 40 E8 8D
1808 40 E8 A9 28 85 01 20 80
1810 18 A9 08 85 01 20 80 18
1818 A9 48 85 01 20 80 18 A9
1820 FD 2D 40 E8 8D 40 E8 A9
1828 F7 2D 21 E8 8D 21 E8 A9
1830 04 0D 40 E8 8D 40 E8 A0
1838 08 20 B0 18 A5 02 9D 01
1840 19 E8 88 D0 F4 A9 FB 2D
1848 40 E8 8D 40 E8 A9 02 0D
1850 40 E8 8D 40 E8 A9 08 0D
1858 21 E8 8D 21 E8 A9 5F 85
1860 01 20 80 18 A9 04 0D 40
1868 E8 8D 40 E8 CE 00 19 D0
1870 91 60 EA EA EA EA EA EA
1878 EA EA EA EA EA EA EA EA
1880 AD 40 E8 29 40 F0 F9 A5
1888 01 49 FF 8D 22 E8 A9 F7
1890 2D 23 E8 8D 23 E8 AD 40
1898 E8 29 01 F0 F9 A9 08 0D
18A0 23 E8 8D 23 E8 A9 FF 8D
18A8 22 E8 60 EA EA EA EA EA
18B0 A9 02 0D 40 E8 8D 40 E8
18B8 AD 40 E8 29 80 D0 F9 AD
18C0 20 E8 49 FF 85 02 A9 FD
18C8 2D 40 E8 8D 40 E8 A9 08
18D0 0D 21 E8 8D 21 E8 AD 40
18D8 E8 29 80 F0 F9 A9 F7 2D
18E0 21 E8 8D 21 E8 A9 FF 8D
18E8 22 E8 60

0001 data to go into bus
0002 data from bus

1900 counter for number of data transfers

1901 start of results area
```

Listing 3. IEEE bus handshake routine in machine language. MLA is My Listen Address; MTA is My Talk Address; UNT is Untalk Command.

read a string. Subroutine 7500 addresses device DV to listen and sends a string. (Note: Routine 7000 reads a string until a carriage return is seen, and then reads one more character. This is because the HP clock ends messages with CR and LF. You might have to change this for your device.)

To reset the clock:
DV = 7:GOSUB 7500
The screen clears and asks for the message (see Example 5).

The Talk Handshake flashes on the screen twice, and the message sent is displayed below:

```
(............)
MESSAGE SENT: R
```

The program uses routine 7000 to read the time. Since DV is already set, we don't have to reassign DV = 7 again. See Example 5. Note that there are three spaces between the colon and the first zero. Two of these are from the HP clock, which starts all messages with two blanks.

The BASIC 488 program, though slow to operate, never times-out and lets you control the IEEE 488 bus. This is helpful when you debug a new IEEE device with your PET.

If you are an experienced 6502 programmer, it is simple to translate the BASIC 488 program into a set of machine-language routines. If you do so, I'd like a copy (tape and source). Listing 3 shows a copy of the IEEE handshakes in machine language. (From the *PET User Notes,* PO Box 371, Montgomeryville, PA 18936, Vol. 1, Issue 7, (Nov.-Dec. '78), p. 8. This is a reprint from the Commodore PET Users Club of England.)

The PET handles the IEEE 488 as a file. Part 2 will cover this. ■

Program Listing Conventions

The PET's graphics and cursor control characters aren't easily duplicated for program listings, so the conventions described here will be used instead.

If a letter or numeral (or any character) is underlined, it means the corresponding graphics character is to be used. (A is the spade symbol on the PET.)

Lowercase letters indicate PET special functions:

clr	Clear Screen	hm	Home Cursor
rt	Cursor Right	lft	Cursor Left
up	Cursor Up	dn	Cursor Down
rvs	RVS field on	off	RVS field off
cr	RETURN key	sp	SPACE key

Sp in a line indicates leading or more-than-one blank. For example, dn/sp/sp/HELLO THERE means Cursor Down space HELLO space THERE.

Two IEEE 488 Instruments

The two instruments described here are typical in the way they are controlled via the IEEE 488 bus. Most instruments are controlled by sending and receiving ASCII characters, which are mnemonics of the function being controlled. For example, the HP clock uses the letter D to increment its days' counter. Numbers are usually sent as ASCII strings—in the same way that PRINT provides an ASCII string of digits to a terminal. CR and LF usually indicate a message's end.

Some instruments will use more difficult formats. Two popular forms are BCD, in which two digits per byte are sent, and pure binary, where the value 0–255 is sent. Be sure you know the *exact* formats used by your instruments! Most instruments are unforgiving of bad data; and the responses range from ignoring meaningless characters to the instrument's unaddressing and leaving the bus. Check your instrument's manual!

The HP 59309A Digital Clock

The HP clock is almost the simplest instrument that uses the IEEE 488 bus. Your options are to either set the time or read the time.

When the clock is addressed to talk, it will provide a string of characters with the time in the following format:
(sp or ?) sp NNDDHHMMSS cr lf
The first character is a space or a question mark. If the clock hasn't been set since the last power-off, the question mark will indicate this. The next two digits indicate the month, from 01 to 12. Then comes the day of the month, 01 to 31. (The clock keeps track of the days in each month correctly and has a leap-year switch). Then the hours (00 to 23), minutes and seconds are sent. The carriage return and line feed indicate the end of the message.

Inside the clock are switches that provide variations of the format—colons or commas can either separate the fields, i.e., NN:DD:HH:MM:SS, or simply send the 24-hour time.

When the clock is addressed to listen, eight ASCII characters are used for control:
P—Stop the clock
T—Start the clock
R—Reset the 01:01:00:00:00
S—Each S will increment the Seconds counter
M—Increment Minutes counter
H—Increment Hours counter
D—Increment Days counter
C—Note time, send it when addressed to talk.

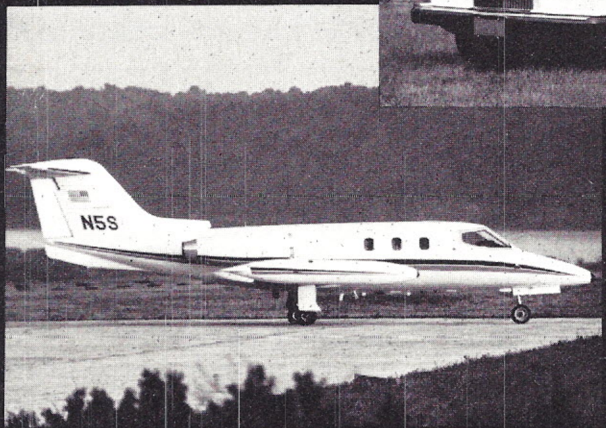
For example, the following string will reset the clock to Jan 5, 8:07:12 AM.
PRDDDDHHHHHHHHMMMMMMMSSSSSSSSSSSST
The T at the end restarts the clock.

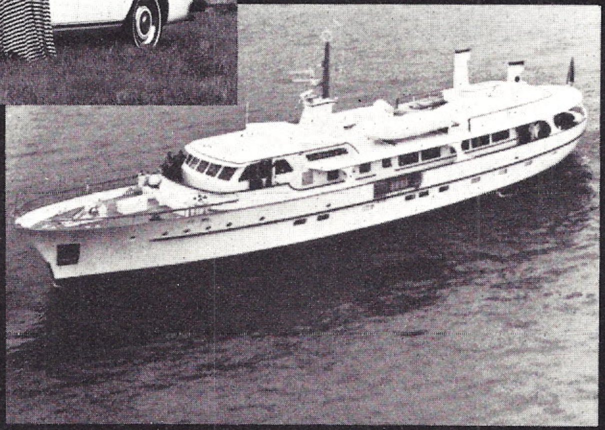
The HP 8165A Programmable Signal Source

This is a "cadillac" 488 instrument—the front panel of this machine has 41 buttons for selection of modes and a 12-button number pad for entering times, and frequencies. This works out to 35 different command formats for setting up parameters and switch settings and nine commands for telling the controller the machine's setting or starting a sequence of actions. Some of the formats include:
F1—Select Sine Wave
F2—Select Triangle Wave
F3—Select Square Wave
FRQ f MZ—Select frequency in MHz. f is a number from 1 to 9999.
FRQ f MZ—Same for Hz
FRQ f KHZ—Same for kHz
SET:—Report all parameters currently operating when addressed to talk.
SET: n—Report setting in memory # n (0–9)

The 8165 can store up to ten complete settings in its memories, so the SET commands permit the controller to find out what's in the 8165.

An instrument of this complexity is usually programmed with a set of special-purpose programs as needed. Writing a general-purpose BASIC program would be both tedious and wasteful. My experience is that the hardest part is to get the PET and the instrument to communicate. Once that is accomplished, the rest is easy.


N5S



New Additions to The Commodore Line

Read all about it: a new computer, new floppies, new software!

Robert W. Baker
15 Windsor Dr.
Atco, NJ 08004

Commodore has announced several new products as part of their PET/CBM line:

- Model 8032 CBM computer—an 80-column CBM with a 12-inch CRT; a new screen editor, version 4.1; and a new disk BASIC, BASIC 4.0.
- Model 8050 dual drive floppy disk—with increased storage capacity.
- Word Pro IV—for the 80-column 8032 CBM.
- ROM upgrade for the 2040 disk operating system—DOS 2.0 to support the new disk BASIC.
- ROM upgrade for the 16 and 32K CBMs to install disk BASIC.

In addition, a CBM modem interface was mentioned recently in one of the trade journals and should be available later this year.

As the author of *Microcomputing's* PET-pourri column, I was fortunate to obtain an early model 8032 CBM from Commodore before writing this article. Although I've only been using the unit for several days, I am impressed. I haven't been able to try the 8050 disk yet. The following information on the 8032 and the new operating software should be enough to whet your appetite.

Acknowledgement
Special thanks to Larry Perry in Commodore Marketing for the equipment and information he provided for the preparation of this article.

The 80-Column 8032 CBM

The 8032 CBM is similar in outside appearance to the former 32K business model CBM. A slight change in the cabinet design tilted the display slightly and changed the keyboard angle. The overall height of the 8032 with its 12-inch CRT is only one inch higher than the older CBM with its 9-inch CRT. I prefer the more comfortable lower keyboard angle.

Besides changing the lower cabinet, Commodore also changed the CRT enclosure to allow much easier access to the CRT controller board. By unscrewing only two screws, you can remove the entire CRT enclosure in one piece giving full access to the printed circuit board.

Internally, the main logic board has been changed slightly, and the metal arm to hold the cabinet open has been moved to the front of the unit. As part of the new logic board layout, the second cassette interface has been moved to the back right side of the PC board. This makes the interface accessible through the side opening of the cabinet where the memory expansion interface was located on the original 8K PET.

As a side effect of this change, the internal expansion connectors are no longer in line with each other. Considering there have been few, if any, products that used these connectors, this change should not cause much concern. The rear cassette interface, user port and IEEE bus connections are still where they always have been.

The CRT controller board has undergone a major change and now has many fewer components. Commodore is now using a 6845 single chip CRT controller, which greatly reduces the time to display a line on the CRT. After doing some simple timing tests, I found up to a two-to-one difference between the older PETs and the new 8032. The 6845 also has many additional features that can be accessed directly via machine language.

The 8032 has the same character sets as the previous CBMs, selected by the familiar POKE 59468,xx command or new control characters. However, the CRT controller now adds two blank rows of dots between

BELL: Sounds the internal bell for ¼ second.	CHR$(7)
DEL LINE: Deletes a full line from the screen, which scrolls up from the point of deletion.	CHR$(21)
ERASE BEGIN: Blanks all characters from the beginning of the line to the current cursor position. The characters do not move over; spaces are inserted in the blanked portion of the line.	CHR$(150)
ERASE END: Blanks all characters from the current cursor position to the end of the line. No characters are moved; spaces are left in the blanked portion of the line.	CHR$(22)
GRAPHIC: Has the same effect as poking 59468 with 12 to select the graphics character set. It also reduces the dot spacing between lines to zero for contiguous graphic characters. Poking 59468 will not affect the dot spacing between lines!	CHR$(142)
INSERT LINE: Inserts a line on the screen by moving all characters below the cursor line, down one line. The bottom line will scroll off the screen and be lost.	CHR$(149)
SCROLL DOWN: Scrolls all lines down one line leaving a blank line at the top of the screen or at the top of the scroll window.	CHR$(153)
SCROLL UP: Scrolls all lines up one line leaving a blank line at the bottom of the screen or at the top of the scroll window.	CHR$(25)
SET BOTTOM: Sets the current cursor position as the bottom right corner of the scrolling window.	CHR$(143)
SET TOP: Sets the current cursor position as the top left corner of the scrolling window.	CHR$(15)
TEXT: Has the same effect as poking 59468 with 14 to select the upper/lowercase character set. It also increases the dot spacing between lines to two for better viewing of text. Poking 59468 will not affect the dot spacing between lines!	CHR$(14)

Table 1. Screen editor control functions will execute the indicated function whenever included as part of a BASIC text string that is being displayed. Each function can be entered as either a control character or by the CHR$(xxx) command.

lines of text when in the upper/lowercase mode for increased visibility.

When in the graphics mode, there are no blank rows of dots between characters to allow true graphics. Display memory still starts at location 32768, but is now 2000 characters long. Programs written for a 40-column CBM that poke display memory will not operate correctly on the 8032.

One other change to the CBM hardware is the addition of an internal "bell." A bell-like sound will be created by software as a sequence of several notes lasting about one-quarter second. The bell is used as an end-of-line warning, such as the bell on a typewriter. It will normally sound in column 75 as the cursor moves to the right through that position. The bell will also be accessible to the software via the ASCII bell code, CHR$(7), in BASIC print commands. The sample unit I've been using doesn't have a bell, but it will be included in the final production units.

In addition to the hardware and mechanical changes, the ROM operating system has been changed. The 8032 CBM includes a new version of the screen editor with many new functions and features. A new disk BASIC includes many of the commands formerly provided by the DOS program (sometimes called the Wedge) loaded from floppy disk.

However, the new BASIC requires a corresponding ROM upgrade in the 2040 disk. The new disk ROMs will also provide several new functions within the disk system, but a new disk format is used. Let's take a look at the new CBM operating system and its improvements before looking at the new disk operating system, DOS 2.0.

8032 Screen Editor

The new version of the screen editor, version 4.1, greatly increases the capabilities of the CBM display. A group of new "control functions" is defined in Table 1. You can request these functions by printing the appropriate control character representing the desired function. These controls let you sound a bell, scroll the display up or down, erase or insert lines, blank out the beginning or end of a line or select the desired character set without poking 59468. Two of the controls, in particular, provide a versatile feature called a "scrolling window."

Normally, the scrolling window consists of the entire screen, 80 columns by 25 lines, and a program can print to any area of the screen. By using the set top and set bottom functions, a scrolling window can be defined as any smaller portion of the entire display area. The scrolling window is defined by positioning the cursor to the upper left corner of the area and printing a set top control character. Then the cursor is positioned at the lower right limit of the scrolling window, and the set bottom control character is printed.

When a scrolling window is defined, the program cannot print outside of the defined area. In addition, when the keyboard home key is pressed, the cursor is returned to the upper left corner of the scrolling window.

The 8032 CBM also has a tab key that will position the cursor at the next defined tab position or at the end of the line if no tabs are defined. If a narrower scrolling window is defined, the cursor will not be advanced past the last column of the scrolling window. The shifted tab key (tab set) is used to set or clear tab stops at the current cursor position.

Remember that the tab and tab set characters can be printed just like any other cursor control character. Even though you shouldn't ever use them at all, you can set up to 80 tabs on the 8032. That's equivalent to setting a tab in every column!

On the 8032, pressing the new ESC key removes the screen editor from the insert or

1. IEEE devices that return a status of one to an INPUT statement would cause the interpreter to hang.
2. The 41st or 81st GET on a line returned garbage. The 256th GET returned the first character of the same line.
3. While key-entering on the last line of the screen (in response to an INPUT statement), the auto scroll up process appended the INPUT statement prompt message to the data read.
4. The 64 millisecond IEEE time-out would result in a status error when a slow device such as a plotter was addressed. Disk BASIC now allows an optional disable of the IEEE time-out by poking 1020 with 0.
5. The software updated the screen display during vertical retrace time. However, the dynamic video RAM on the newer PETs can be updated during non-vertical retrace times without glitching the vertical retrace.
6. The TI$ function in the business keyboard version lost time when the screen scrolled. TI$ is now accurate.
7. Inputting more than 80 characters without a RETURN caused a system crash. Disk BASIC generates a ?STRING TOO LONG ERROR message.

Table 2. Disk BASIC enhancements and corrections to correct errors in version 2 BASIC, in addition to those mentioned in the text.

quote modes. Suppose you're typing the text for a BASIC print statement following the quote and discover a mistake further back in the line. On the older model CBMs you would have to delete all the characters back to the one in error, or type another quote to exit quote mode and allow cursor controls.

On the 8032 you simply type ESC to exit quote mode and allow the cursor controls. You then go back and correct the character in error. After repositioning, you can continue the line of text. However, you cannot enter a control key now, since you are not in quote mode.

This feature also allows you to enter control characters without using the CHR$(xxx) command. Any characters entered in reverse field are turned into control characters in the string by BASIC. Thus, typing quote-ESC-RVS-g enters a control-G or bell code (07). The ESC key allows the RVS key to function instead of being entered in the string.

One other keyboard control that's been changed is the run key, which now enters a DLOAD-RETURN-RUN-RETURN sequence. This command sequence will load and run the first program found on the floppy disk in drive #0 on the 2040. You can, however, still load programs from cassette tape by simply typing the former LOAD command.

Disk BASIC

The first Commodore BASIC 1.0 was released in August 1977 for the PET 2001-8 computer. Version 2.0, released in July 1978, added a machine-language monitor and corrected the bugs of version 1.0. In May of 1979, version 3.0 was completed but never released for production. This version was intended to enhance the speed of string processing.

The new version 4.0 of Commodore PET BASIC, referred to as disk BASIC, includes all enhancements of the previous releases plus several new improvements as shown in Table 2. In addition, the 2040 disk commands have been integrated into the BASIC language, and a new random access file structure has been implemented.

Disk BASIC is the standard BASIC in the new 8032 CBM and can also be added as a ROM upgrade to the older CBMs. It does, however, require a corresponding new ROM set for the 2040 floppy disk system, DOS 2.0.

One enhancement contained in this latest ROM set is improved performance of string garbage collection. Garbage strings are any strings that occupy memory space but are no longer referenced by BASIC. To allocate a new string, the string storage must occasionally be compressed by eliminating these garbage strings.

In version 2 of the PET BASIC ROMs, the algorithm to compress this space is extremely slow, often requiring several minutes to complete. To the user, the PET appears dead with no response from the keyboard or the stop key. Disk BASIC uses a new, much faster, algorithm that requires less than a second, even in the worst case.

The 14 new disk commands are shown in Table 3 with a brief description of each command. The parameters for each disk command are not order dependent, e.g., the file name, drive number and unit number can be specified in any order. File names may be in quotes or represented as a string variable. Drive numbers are indicated by the letter D, followed by 0 or 1. The default drive number is generally 0 but may be the last user-specified drive number.

The unit number (IEEE bus address) is optional on nearly all commands and defaults to 8. A user-specified unit is indicated by the letter U, followed by an integer between 4 and 31. Variables or expressions to be evaluated in disk command parameter lists must be enclosed in parentheses, e.g., U(D) or U(2 + B). The second cassette cannot be used simultaneously with the new PET disk commands.

With the addition of the disk commands, two disk status variables, DS and DS$, have been defined. These reserved variables cannot be assigned a value by the user or a BASIC program. Whenever they are referenced in a print statement or the right-hand side of an expression, the disk command channel will be queried and the updated values assigned. The operating system will keep a flag to prevent rereading these values if a subsequent disk operation has not been performed.

With Disk BASIC and the corresponding DOS 2.0, you no longer have to bother with the disk error and control channel, channel 15. All error information is returned automatically simply by referencing the disk status variables.

For added convenience when writing sequential disk files, the PRINT# command has been modified in Disk BASIC. A PRINT# command will no longer transmit a line feed, CHR$(10), after a carriage return if the logical address is less than 128. Thus, you no longer have to write:

```
PRINT#8,A$;CHR$(13);
```

Instead, you simply write

```
PRINT#8,A$
```

If needed, all logical addresses greater than 128 will still send a carriage return and line

DOPEN—opens a disk file in the mode specified.

DCLOSE—can close all files currently open on a disk unit or only the logical file specified.

RECORD—is executed directly before GET#, INPUT# or PRINT# to position the disk at the desired record when using random access files. If this command is not executed, then the position accessed will be the next record directly after the last referenced record. The relative record number must be specified between 1 and 65535. An optional character position within a record can be specified between 1 and 254.

HEADER—formats the disk specified, when a disk ID number is specified. Otherwise, the directory is cleared and a new name is assigned to the disk. In direct mode, BASIC prints the following prompt and begins to flash the cursor:

ARE YOU SURE?

The command will not execute without a Y-RETURN or YES-RETURN response from the user.

COLLECT—frees up allocated space of improperly closed files on the disk and deletes their references from the directory. The block allocation map (BAM) stored on the disk is also verified.

BACKUP—duplicates a disk complete with disk name, ID, file layout and contents. It can only be used on version 2-formatted diskettes.

COPY—will only function on drives within a single floppy disk unit. It can create a copy of a file within the same drive or on the other drive in the unit. Copy without file names copies all files from one disk to another without altering files that already exist. Any files that have common names will cause a #63 error when copying all files.

CONCAT—concatenates two disk files.

DSAVE—saves a BASIC text file on disk.

DLOAD—loads a BASIC text file from disk.

DIRECTORY—displays the directory of a diskette. A printer can be addressed by CMD so that this command will produce a hard-copy directory listing. If a specific drive number is given, DOS 2.1 will initialize the drive.

RENAME—changes the name of a file on diskette.

SCRATCH—deletes a file from the disk directory, effectively deleting the file. The ARE YOU SURE? prompt will be given in direct mode as with the HEADER command. Any open files cannot be scratched.

APPEND—similar to a DOPEN command, except it applies only to sequential files. The 2040 write pointers are positioned to the next character position beyond the end of file. Additional data may then be written and the file re-closed.

Table 3. Disk BASIC disk commands.

feed at the end of each line, just as before.

Companion Disk Operating System

When upgrading to disk BASIC, you must install a corresponding ROM upgrade in the 2040 disk system. This ROM upgrade for the 2040 floppy-disk system can, however, still be used with the older version BASIC without any problems, except the lack of disk support directly within BASIC. The new disk operating system, DOS 2.0, has many changes besides the addition of true random access disk files. Before examining the new random access file structure, I will discuss the other changes made to DOS 1.2.

The most significant change is in the actual disk format, where a change was made to improve reliability in read and write operations. Diskettes created by either version 1 or 2 can be read by DOS 2.0, but only diskettes formatted by DOS 2.0 can be written by DOS 2.0. Thus, you must convert all diskettes to the new format if you intend writing to the diskette or want to back up version 1 diskettes with DOS 2.

The new disk format contains one less sector on tracks 18 through 24. This limits the number of directory entries to 144 and removes six data blocks from the diskette. Since the duplicate (BACKUP) command is a block-for-block copy, version 1 diskettes cannot be backed up with the BACKUP command on DOS 2.0. A new disk-to-disk copy command has been provided as a built-in utility for converting to the new format, as well as for general backup operations.

The following commands cannot be used with the old 2040-formatted diskettes (DOS 1.2):

DOPEN (write operations)
HEADER (without IZZ)
COLLECT
BACKUP
COPY (writing new files)
CONCAT
DSAVE
RENAME
SCRATCH
APPEND

These commands can be used:

INITIALIZE
COPY (reading files)
DLOAD
DIRECTORY

For added convenience, the DOS version and format type will be indicated in the directory heading whenever the DIRECTORY command is issued. When 2A is displayed, it indicates that version 2 DOS formatted the diskette with 2040 format (Shugart 390 drives). Future format types will be indicated by different prefix letters other than A. When 1 appears in the directory, the diskette has the format associated with version 1 DOS.

The version 1 DOS will not generate an error if a write operation is attempted on an A-formatted diskette. However, the headers will be disturbed, and successive operations will cause errors. If a write operation is attempted on an old 2040-formatted diskette by DOS 2.0, a 73 error code will be generated: 73,CBM DOS V2, track, sector.

Another major change incorporated into DOS 2.0 is an auto-initialization function. DOS 2.0 will manage the diskette initialization process prior to any file command or OPEN file sequence. The DOS compares the diskette ID from a header on the directory track against a master copy in memory. If they do not match, an initialization process will be invoked. This applies only to functions that involve directory searches. This new feature eliminates the initialize command required by DOS 1.2 every time you insert a diskette.

However, it is now important to assign a unique ID to each diskette. The auto-initialization function depends on a change of ID to detect a change of diskette. The original initialization command is still supported by DOS 2.0 (but not by Disk BASIC) for current software and explicit initialization. If required, the auto-initialization function can be switched off by:

```
OPEN 1,8,15
PRINT#1, "M-W" CHR$(243);CHR$(16);CHR$(1);CHR$(1)
```

The other areas changed involve disk and file copying functions. Disk copying via the BACKUP command is now faster and more reliable. An error counter has been added to abort the function if a predetermined number of errors are reached. In addition, the back-up process now takes about 2 minutes and 15 seconds, as opposed to 6 or 7 minutes in the old system.

The file copy command has been expanded to include a copy of all files from one disk to another disk, as previously provided by a BASIC utility program. Other special copy commands have been added to concatenate or append files as desired.

A few errors have been found and documented in the new DOS 2.0. The most serious or most often seen errors include:

•The save-replace problem originally found in DOS 1.2 still exists.
•Scratch will not remove a file recently used but properly closed. The system may have to be reset before allowing the file to be scratched.
•A bad status (ST) may be returned when DS$ is accessed following a disk operation.
•Pattern matching with trailing question marks does not function properly. The compare routine in the file-name search matches characters on a character-by-character basis. If the file name is smaller in length than the length of the pattern without the trailing question marks, a match is made improperly. For example, A??? will match A, AA, AAA and AAAA, but not AAAAA.
•Relative files cannot be copied with the COPY command.
•The relative record pointer may not be handled properly in all cases following a PRINT#. A RECORD command should be issued before each PRINT# command.

Overall, the entire disk operating system has been greatly enhanced and improved over the previous version, even without the addition of random access files.

Relative Record File System

Ever since the first 2040 floppy disks were released, many people have complained about the lack of a good random access file structure. This shortcoming of the early systems made many applications much more complex than necessary. As a result, many people avoided using random access files where they really would have enhanced a particular program package.

The new DOS 2.0 has taken care of that by including an advanced random access file structure with all the necessary support functions. The overall relative record file system is much like those of many larger systems and easy to use.

Relative files involve a direct access method that allows the programmer to position to any record relative to the beginning of the file. Record sizes are fixed in length and may range from 1 to 254 bytes per record. Record numbers are limited to the capacity of the disk but may not exceed 65535.

The two main components of a relative file are the side sector chain of blocks and the data block chain. Both are linked together through forward pointers similar to a sequential file.

Data block pointers in the side sectors allow the DOS to move from one record to another within two disk reads. The side sector also contains a table of pointers to all of the other side sectors within the file. To move from one side sector to the other, the pointer is referenced and the corresponding track and sector is read. Once the proper side sector is read into memory, the data block pointer is referenced, and that track and sector is used to read in the actual data block containing the record.

A file may contain up to six side sectors, each of which may contain pointers to 120 data blocks. Therefore, the largest file on the 2040 DOS 2.0 would be:

120 data block pointers per side sector
× 6 side sectors
× 254 bytes per data block
= 182,880 bytes, which is greater than the current capacity of the 2040 disk format.

The side sectors do not contain record information, but contain the locations of the data blocks. The record size is used to compute the locations of the data blocks and

Model 8050 dual drive floppy disk from Commodore.

where the pointer is placed when a record number is given through the RECORD command.

To open a relative record file for the first time, the DOS generates one data block and one side sector. For a record size of 10, 25 records will be generated automatically upon the opening using 250 bytes of the available 254 data bytes. Then you expand the file by a reference to the last record number you wish to generate through the RECORD command that prints it. Intermediate records from the current end to the referenced record number will be generated by the DOS. Any side sectors and all data blocks necessary to contain a file of this size will also be generated.

The relative channel requires three memory buffers from the system; whereas, sequential files only require two. Since there are only 12 buffers in the system, and two of these are used in directory searches and internal functions, only three relative channels can be open at one time. Any combination of sequential and relative record files can be open at one time as long as no more than ten buffers are required.

Every time the DOS positions to a record—either through the RECORD command or a previous I/O statement to a record—it performs a read-ahead function for further sequential action. This is transparent to the user, since the 2040 performs all these I/O functions independent of the main CPU.

In the case where a read-ahead record spans two data blocks (starts in one data block and ends in another data block), the DOS simply reads in the remaining part of the record, as well as any following records in the next data block. The records of most relative files will span across data blocks, since only record sizes of 1, 2, 127 and 254 divide evenly into the 254-byte size of a data block.

When a record is written into through the PRINT# statement, the data block is not immediately written out to the disk. It is only written to the disk when the DOS moves beyond the particular data block that the record resides in. This can occur through successive printing to sequential records or positioning to another record outside of that particular data block. This feature reduces the number of disk reads and writes and will improve time efficiency.

However, because of this feature, it is recommended that two channels not be open to a relative file at one time if either channel will be writing to the relative file. An update may be made in the channel's particular memory buffer area without causing a change on the disk file or the other channel's buffer area. There is no restriction if the channels are simply reading from the file. It may actually be advantageous for certain applications to have more than one channel open to a single relative file.

The relative file has also been designed so that you don't have to specify record numbers when printing to it. The relative file can be treated as a sequential file of certain record lengths. Whenever a write or read operation through an INPUT# or PRINT# statement is performed, the next reference record will be the sequential record. This feature can be used to simulate variable record size in a relative file by using a small record length such as ten and referring to the beginning of a variable record and N number of records thereafter. Each of these groups could be referred to as a logical record.

The DOS terminates printing to a record by sending the EOI signal from the IEEE. This signal is generated on every PRINT# statement regardless of its form. Therefore, each relative record must be printed with a single PRINT# statement. If the PRINT# statement goes over the preset record size, error #51,RECORD OVERFLOW will be generated. The information will be truncated to the number of characters specified by the record size, and the DOS will position to the next record in sequence.

If the PRINT# statement contains fewer characters than the record size, the remaining positions within the record will be filled with nulls (0). If you need to store binary information, you will have to use a record terminator such as a carriage return, CHR$(13), and increase the record size by one to accommodate the terminator.

When generating new data blocks for relative files, the requested record number is checked against the number of data blocks left on the diskette. When the resulting number of data blocks is greater than what is left on the diskette, error #52,FILE TOO LARGE is generated. This protects you from generating a file with an error and requiring them to scratch the file to recover the space.

Review

The introduction of the 8032 CBM could be the most significant move by Commodore to increase their share of the current small-systems market. The 8032 has all the proven features of the earlier PET/CBM models, plus a more common display size and an enhanced screen editor. If it's as reliable as previous models, it could easily become the most popular system in the near future.

In addition, if the preliminary documentation I've seen is any indication of what to expect with future products, the complaints about poor documentation from Commodore should quickly disappear. If you're ready to buy a new system, you should take a look at the 8032 when it becomes available in your area.

The only problem I see with the new 8032 CBM is with support software. For anyone to be able to support every PET/CBM model and operating system is almost impossible. Now, more than ever, you'll have to be certain that desired software will run on your particular system.

When ordering software, indicate as much information about your system as possible. Be sure to indicate the correct model and any ROM upgrades added since received. I hope someone will come up with an easy way to identify the various com-

binations that will be possible. Until that time, the more information supplied with an order, the better your chances the programs will run on your system.

Almost any program written for the older models must be modified to run on the 8032. Remember that Disk BASIC is a new ROM release. The routines in the ROM operating system and possibly low memory pointers have been changed again, just as in the last ROM release. In addition, the screen size is now twice as wide, and most programs will not handle the display properly, especially if you poke screen memory, rather than use BASIC PRINT statements.

If you already own a 16 or 32K PET/CBM, remember that you can upgrade to Disk BASIC by simply changing your ROMs. You won't have 80 characters per line, but you will have the new ROM operating system. However, you will have the same compatibility problems as new 8032 owners have, except for the differences caused by the 80-column display.

For example, Commodore's Word Pro 3 word-processing package and Eastern House Software's MAE assembler package will not run with Disk BASIC. It appears that owners of the 8K PETs will be quickly left behind. I don't believe that Disk BASIC will be available for the 8K PETs, but I haven't seen any final word on this.

Model 2040 floppy disk owners can also upgrade their ROM sets to take advantage of DOS 2.0 and its related features. You don't have to have Disk BASIC to use DOS 2.0, but I'm not sure if they will be available separately. Keep in mind that you will most likely have to recopy every diskette in the new disk format. This can be a major task if you have a number of disks.

In my case, it took almost a week to convert every disk to the new format. In the long run, however, I feel it was well worth it. I've had much fewer disk problems, and most disk operations seem to run faster. Once you change the ROMs, though, you can only read old-format disks. ■

UPI

PASCAL

FORTRAN

APL

STOCK QUOTES

TRS-80™ Super Version #1

Why limit yourself to Radio Shack® BASIC or spend hundreds of dollars on new programs and languages?

Through the phone lines, your TRS-80™ becomes a sophisticated and powerful computer. Access new languages, make vital decisions based on stock quote, earn money selling those top notch programs you write.

Two services, MicroNet™ and The Source™ open the world to the 80 user. Pick the group for your needs or both!!

MicroNet
- Hookup $9.00
- Per hour $5.00

The Source
- Hookup $100.00
- Per hour .... $ 2.75

These charges are billed to your Master Charge or VISA.

Houston Micro Computer Technologies, Inc. offers you three packages to choose from:

Package I (ST80III, Accoustic Modem, RS232C) with:
- MicroNet .... $460.
- The Source .... $540.
- Both .... $545.

Package II (ST80I, Accoustic Modem, RS232C) with:
- MicroNet .... $400.
- The Source .... $480.
- Both .... $485.

Package III for Mod II (HMCTERM, Accoustic Modem, Cable)
- MicroNet .... $500.
- The Source .... $580.
- Both .... $585.

ST-80 I ($25.00)
Set Serial Board from Keyboard
Auto Line Feed
Line Printer on

ST-80 II ($85.00)
Dump Memory to Disk
Allow Lower Case Letters
Transmit Auto Logon
Transmit Buffer to Serial: Out Port
Serial: Into Memory

ST-80 III ($150.00)
Help Command
Auto Answer Auto Originate
Switch Status
Warm Program Restart
Job Log

HMCTERM ($200.00)
Converts TRS-80 Model II to Complete Intelligent Terminal for Processing with Major Mainframes

---

To order by phone or for local dealer information call: **713/661-2005**
Texas residents add 6% sales tax • MasterCharge • Visa

HOUSTON MICRO-COMPUTER TECHNOLOGIES, INC.

Home and Business Computer Specialists

5313 BISSONNET • BELLAIRE • TEXAS • 77401 • 713/661-2005

Dealer inquiries invited.

33

A File Sorting Program and Its Diary

This article's conclusion completes the sorting routine and further explores Apple DOS.

Al Prentice
PO Box 2998
Escondido, CA 92025

Last month, in part 1 of the article, we began writing a file sorting program for the Apple II with disk drive. In the process, we examined the kind of planning that should precede the writing of this or any other relatively complex program. Our preparation completed, we began by writing the main or "top level" program and the first two subroutines. And we learned a fair amount about Apple DOS while doing so. In this concluding installment, we'll complete the sorting program and discover still more about Apple DOS.

Extracting the Key Fields

The subroutine that extracts the key fields from the input file and records their original positions is the next programming task to be tackled. I approached this task by studying the file layout (see Fig. 5, part 1) and, as I did for the top level program, wrote a sketch of the subroutine[1] (see Example 1). I then used this sketch as an outline to write the subroutine in Fig. 1.

Two new aspects of the use of Apple DOS arise in this subroutine. First consider line 2120. If HS = 3 and RG = 14 then the command:

READ CHAOTIC MESS,R3

will be issued when K = 0, and the command:

READ CHAOTIC MESS,R17

will be given when K = 1. A READ command is used to tell Apple DOS that we wish to read (as opposed to, say, write) a given file. In the case of fixed-record-length files, the R parameter specifies which record is to be read next.

In Apple DOS the READ command does *not* cause any data to be read. It just informs Apple DOS that we are going to read data from the named file. Therefore, neither statement 2120 nor 2150 reads any data from the disk. The data is read by statements 2130 and 2160.

The way that Apple DOS works with both BASIC programming systems is that a READ command says, "From now until we turn off the read, all data that would come from the keyboard to an INPUT statement should come from the named file." This is the second important feature of Apple DOS.

Using the same trick that I used to check out the routines at lines 4000 and 6000 (i.e., setting D$ = "%"), we can check out a program which reads data from a file without having any data filed on the disk! If the first character of the message part of a PRINT statement is not a control-D, then Apple DOS will ignore it. Therefore, if we assign the variable D$ the value "%", what would have been READ commands to DOS are just messages printed on the screen, and subsequent INPUT statements (which would have read data from the disk) are just INPUT statements that expect data from the keyboard! We can simulate the disk from the keyboard while we check out our programs.

This is a useful feature because it is often much easier to simply enter the kinds of data you want for debugging than it would be to make up files with all of the kinds of data needed for checkout. I used this technique to debug the subroutine at line 2000.

Before I go on to discuss the debugging run, I should point out one aspect of the READ command which is issued by the statement at line 2150. The last few characters (;",B";KF) cause a B parameter to be included in the READ command. If KF is currently 7, then the READ command will end with ,B7.

The B parameter in a READ command tells DOS that some of the initial characters (or bytes) in the record are not to be input. For example, B7 means that we wish to start reading the record with byte number 7. I used this feature of the READ command to skip over the bytes in the record which precede the key field. Now let's examine the debugging run.

The results of the debugging run are shown in Fig. 2. With D$

```
k = 0
fi = false
DO WHILE fi = false
    READ record-number hs + (k*rg)
    IF not end-of-file-string
        THEN READ record-number hs + (k*rg) + kr into work-string
            key = k1 characters beginning with character kf in work-string
            index(k) = k
            k = k + 1
        ELSE fi = true
END DO WHILE
RETURN
```

Example 1.

[1]In the jargon of modern programming, writing these sketches is called using a program design language to design the program. There are formal program design languages which are used effectively in the design of large software systems. They are machine readable, and there are processors that read those designs and produce useful design reports and maintenance documentation. We are using an ad hoc, informal program design language for designing the routines in the file sort routine.

```
2000  REM  --------------------
2010  REM  EXTRACT KEY FIELDS
2020  REM  --------------------
2100 K = 0
2110 FI = 0
2120  PRINT D$;"READ ";OF$;",R";HS + K * RG
2130  INPUT "1ST RECORD? ";WK$
2140  IF  LEFT$ (WK$, LEN (EF$)) = EF$ THEN  GOTO 2500
2150  PRINT D$;"READ ";OF$;",R";HS + K * RG + KR;",B";KF
2160  INPUT "KEY FIELD? ";WK$
2170 KY$(K) =  LEFT$ (WK$,KL)
2180 IX(K) = K
2190 K = K + 1
2200  GOTO 2600
2500 FI = 1
2600  IF FI = 0 THEN  GOTO 2120
2990  RETURN
```

Fig. 1. Reads the file to be sorted and extracts the key field from each group. As it does, it checks the first record of each group to see if it contains the end-of-file signal.

set to % rather than a control-D, the execution of the routine at 6000 just prints the message:

"%OPEN CHAOTIC MESS,L41"

but DOS ignores it, and the file is not opened. The execution of the routine at line 2000 similarly prints the message:

"%READ CHAOTIC MESS,R3"

Since I had said that the file had a header which was three records long, this was the correct command to be issued.

In order to understand why this is so, two things must be known about the READ command. First, when we are dealing with fixed-record-length files (and have opened them with an OPEN command containing an L parameter), the R parameter specifies *which* record is to be read by the next INPUT statement. This is called direct access, because in this case DOS will not just give us the next record from the file but will go directly to the specified record and access it. Thus, the READ command shown above says to DOS, "the next INPUT statement should receive the data from record number 3 (of the file currently open for reading) when it is executed."

It might appear that this would result in skipping over only two records (although the header was specified to be three records long!), but, in fact, it is skipping the first three records. DOS considers records to be numbered 0, 1, 2, . . . rather than 1, 2, 3, . . . , and so, reading record number 3 skips records numbered 0, 1 and 2.

In the debugging run, the program after printing the READ command asked for input (1ST RECORD?). Had the program actually opened the file and issued the READ command to DOS, the INPUT statement would have received its input from the file. However, in this debugging run, statement 2130 was just a normal INPUT statement that asked for data from the keyboard.

I responded on the keyboard by typing "NOT AN EOF." Since this simulated record did not match the end-of-file signal which I had specified, the program went on to read the record containing the key field. It issued the command "% READ CHAOTIC MESS,R6,B7," setting up the next INPUT statement to read from record number six starting with the byte numbered seven.

Studying this message, I discovered two errors! The routine should have been reading the sixth record starting with the seventh byte, but it was asking for the seventh record starting at the eighth byte! So, I made a note to change line 2150 to:

```
2150 PRINT D$;"READ";OF$;",R";HS + K
     * RG + KR - 1;",B";KF - 1
```

The program then executed line 2160, and I pretended to be the disk by typing:

7890123456789012345678901234567890

The routine stored the key and asked again for the 1st record (meaning the 1st record of a record group). This time it was asking to read record number 17 (the 18th record of the file), which was correct for the file, which was specified to have a 3-record header and record groups consisting of 14 records.

I responded with "END-OF-FILE-MARK," which was the end-of-file signal that I had specified for the file. The routine correctly recognized the end-of-file signal and exited. The program called the routines at lines 1000 and 3000 and then the routine at line 4000. That routine printed the correct CLOSE command and exited to the top level program, which executed the END statement.

At this point, I had discovered the two errors in statement 2150, but I had not yet checked out the part of the routine that stored the keys.

Since I had an appropriate kind of file available (see Fig. 1a, part 1) and the program seemed to be working well, I decided to begin using DOS and real files for my checkout runs. I corrected statement 2150, changed statement 390 to set D$ to a control-D and changed line 580 to go to the following debugging routine, which I added at line 10000:

```
10000 FOR K = 0 TO 10
10010 PRINT K;" ";IX(K);" ";KY$(K)
10020 NEXT K
10030 END
```

My next debugging run gave the appearance of working correctly, except that the program did not recognize the end-of-file

```
]RUN
WHAT IS THE NAME OF THE FILE TO BE
SORTED? CHAOTIC MESS
WHAT IS TO BE THE NAME OF THE SORTED
FILE? A GOOD SORT

THE INPUT FILE MUST BE A FIXED RECORD
LENGTH FILE.



WHAT IS THE LENGTH, IN CHARACTERS, OF
THE RECORDS (EXCLUDING THE FINAL <CR>)? 40
     .
     .
     .
HOW MANY RECORDS ARE IN A RECORD GROUP
IN THE INPUT FILE? 14
     .
     .
     .
WHICH RECORD, WITHIN A RECORD GROUP
CONTAINS THE KEY FIELD? 3

WHICH CHARACTER, WITHIN THAT RECORD IS
THE FIRST CHARACTER OF THE KEY FIELD?   7

HOW LONG, IN CHARACTERS, IS THE KEY
FIELD? 25
     .
     .
     .
HOW LONG, IN RECORDS, IS THE HEADER ON
THE INPUT FILE? 3
EACH FILE MUST END WITH AN 'END-OF-FILE
RECORD.  THIS IS A RECORD WHICH DOESN'T
GET SORTED BUT CONTAINS A STRING OF
CHARACTERS WHICH IS UNIQUE.

WHAT DOES THE EOF RECORD CONTAIN? END-OF-FILE-MARK
%OPEN CHAOTIC MESS,L41
%READ CHAOTIC MESS,R3
1ST RECORD? NOT AN EOF
%READ CHAOTIC MESS,R6,B7
KEY FIELD? 7890123456789012345678901234567890
%READ CHAOTIC MESS,R17
1ST RECORD? END-OF-FILE-MARK
     .
     .
     .
```

Fig. 2. Sample run. Illustrates a versatile debugging technique that can be used with Apple DOS. The string variable D$ has been assigned the value %, which allows simulating the actions of the disk by making entries at the keyboard. The lines that begin with the character % show the commands that would have been issued to DOS. Input statements, which would have read data from the disk file, are executed in the normal manner and expect to receive data entered at the keyboard.

```
]RUN
WHAT IS THE NAME OF THE FILE TO BE
SORTED? PEOPLE
WHAT IS TO BE THE NAME OF THE SORTED
FILE? PEOPLE INDEX
WHAT IS THE LENGTH, IN CHARACTERS, OF
THE RECORDS (EXCLUDING THE FINAL <CR>)? 20

HOW MANY RECORDS ARE IN A RECORD GROUP
IN THE INPUT FILE? 3

WHICH RECORD, WITHIN A RECORD GROUP
CONTAINS THE KEY FIELD? 3

WHICH CHARACTER, WITHIN THAT RECORD IS
THE FIRST CHARACTER OF THE KEY FIELD?   13

HOW LONG, IN CHARACTERS, IS THE KEY
FIELD? 8

HOW LONG, IN RECORDS, IS THE HEADER ON
THE INPUT FILE? 6

WHAT DOES THE EOF RECORD CONTAIN? END-OF-FILE
1ST RECORD? BLOW JOE
KEY FIELD? 1ST RECORD? B.
KEY FIELD? 1ST RECORD?
KEY FIELD? 1ST RECORD?
KEY FIELD? 1ST RECORD? EN.
KEY FIELD? 1ST RECORD? STER
KEY FIELD? 1ST RECORD?
KEY FIELD? 1ST RECORD?
***DISK: END OF DATA ERROR

BREAK IN 2130
]GOTO 10000
0 0 CLASS 1A
1 1 CLASS 1B
2 2 CLASS 4A
3 3 CLASS 3A
4 4 CLASS 2A
5 5 CLASS 1A
6 6
7 0
```

Fig. 3. The first debugging run to use real data from the disk and to invoke Apple DOS reveals a problem caused by a bug in Apple DOS. The file used as input is shown in Fig. 1a (part 1). You should compare the responses given during this run with the format shown in Fig. 1b (part 1).

signal and attempted to read records past the end of the data in the file. When this occurs DOS prints the message:

***DISK: END OF DATA ERROR

and causes a break in the program.

Since this part of the program had appeared to work correctly when I was simulating the disk from the keyboard, I knew that if the program was getting the correct data from the disk it would recognize the end-of-file signal. Furthermore, the READ commands which were printed in the earlier run had included the correct record number for the first record of the record groups.

In order to find out what was happening, I added a statement after line 2130 to print out the contents of WK$ and ran the run again. This run is shown in Fig. 3. The program had indeed gotten the correct data for the first time it read the first record of the record group ("BLOW JOE . . ."), but all of the subsequent tries had produced some garbage. A little study revealed that, after the first time, statement 2130 had input data starting with the 13th character of the first record of each record group! This was caused by the only bug (which shows what a good job Apple did on DOS) that I have found in DOS.

When the B parameter is used in a READ command to DOS it remains set until superceded by a B parameter in another READ command. Furthermore, once it has been set it is not reset to 0 even by RUN commands! Therefore, statement 2130 got the correct data the first time it was executed, but statement 2150 set the B parameter to 13, and all subsequent executions of 2130 started with data at the 13th character of the record.

I made a note to add ";B0" to the end of statement 2120. Then, in order to see if the key extraction was working correctly, I punched in "GOTO 10000." The results, shown at the bottom of Fig. 3, proved that the key fields and their record-group numbers were being extracted correctly. In order to thoroughly exercise the key extraction routine, I made several debugging runs whose results are shown in Figs. 4a through 4d.

a.
```
1ST RECORD? BLOW JOE
KEY FIELD? 1ST RECORD? NIMBLE JACK B.
KEY FIELD? 1ST RECORD? CINDERELLA
KEY FIELD? 1ST RECORD? KENT CLARK
KEY FIELD? 1ST RECORD? BULLWINKLE GEN.
KEY FIELD? 1ST RECORD? CROWLEY ALEISTER
KEY FIELD? 1ST RECORD? END-OF-FILE
AT ROUTINE 1000
AT ROUTINE 3000
0 0 CLASS 1A
1 1 CLASS 1B
2 2 CLASS 4A
3 3 CLASS 3A
4 4 CLASS 2A
5 5 CLASS 1A
6 0
```

b.
```
1ST RECORD? BLOW JOE
KEY FIELD? 1ST RECORD? NIMBLE JACK B.
KEY FIELD? 1ST RECORD? CINDERELLA
KEY FIELD? 1ST RECORD? KENT CLARK
KEY FIELD? 1ST RECORD? BULLWINKLE GEN.
KEY FIELD? 1ST RECORD? CROWLEY ALEISTER
KEY FIELD? 1ST RECORD? END-OF-FILE
AT ROUTINE 1000
AT ROUTINE 3000
0 0 CHICAGO
1 1 CANDLESTICK PARK
2 2 SHANTYTOWN
3 3 METROPOLIS
4 4 NORTH WOODS
5 5 HIGHER PLANES
6 0
```

c.
```
1ST RECORD? BLOW JOE
KEY FIELD? 1ST RECORD? NIMBLE JACK B.
KEY FIELD? 1ST RECORD? CINDERELLA
KEY FIELD? 1ST RECORD? KENT CLARK
KEY FIELD? 1ST RECORD? BULLWINKLE GEN.
KEY FIELD? 1ST RECORD? CROWLEY ALEISTER
KEY FIELD? 1ST RECORD? END-OF-FILE
AT ROUTINE 1000
AT ROUTINE 3000
0 0 BLOW JOE
1 1 NIMBLE JACK B.
2 2 CINDERELLA
3 3 KENT CLARK
4 4 BULLWINKLE GEN.
5 5 CROWLEY ALEISTER
6 0
```

d.
```
1ST RECORD? BLOW JOE
KEY FIELD? 1ST RECORD? NIMBLE JACK B.
KEY FIELD? 1ST RECORD? CINDERELLA
KEY FIELD? 1ST RECORD? KENT CLARK
KEY FIELD? 1ST RECORD? BULLWINKLE GEN.
KEY FIELD? 1ST RECORD? CROWLEY ALEISTER
KEY FIELD? 1ST RECORD? END-OF-FILE
AT ROUTINE 1000
AT ROUTINE 3000
0 0 NDBA
1 1 GH J
2 2 LDIG
3 3 RONG
4 4 MPAN
5 5 GICI
6 0
```

Fig. 4. (a) Key field is eight characters long, starting with the 13th character of the third record. (b) Key field is 20 characters long, starting with the first character of the second record. (c) Key field is 20 characters long, starting with the first character of the first record. (d) Key field is four characters long, starting with the third character of the third record.

An Efficiency Improvement

I had known from the start that there was a special case which the key extraction routine (as specified) would not take advantage of: If the key field was in the first record of a record group (which would be true for all of those files which had only one record in a record group, and frequently true for others as well), the routine would read the first record twice! It would read it first when looking for the end-of-file signal, and again to get the record which contained the key.

I had been following a programming practice that I use when working with interactive programming systems or a personal computer. I first program the routine and check it out without including any of the code to handle special cases, particularly those that only affect the efficiency of the routine. Once I have that version of the routine working, I add the code to handle the special cases and debug it again.

When the special cases deal only with efficiency issues, the subsequent debugging is particularly easy since the routine should give exactly the same results after the new code is added.

In the case of the key extraction routine I added the following three statements:

```
2145 IF KR<>1 THEN GOTO 2150
2147 KY$(K) = MID$ (WK$,KF,KL)
2149 GOTO 2180
```

I then ran the same tests again and got the same results as from the previous runs.

```
1000  REM  ----------------------
1010  REM  SHELL SORT FOR STRINGS
1020  REM  ----------------------
1021  PRINT : PRINT "----------------"
1100 GA =  INT (K / 2)
1110 I = GA + 1
1120 J = I - GA
1130 JG = J + GA
1131  PRINT "<"; LEFT$ (KY$(J),6);"> <"; LEFT$ (KY$(JG),6);">";
1132  PRINT "GA=";GA;" I=";I;" J=";J;" JG=";JG
1140  IF KY$(J) <  = KY$(JG) THEN  GOTO 1400
1150 WK$ = KY$(J)
1155 W = IX(J)
1160 KY$(J) = KY$(JG)
1165 IX(J) = IX(JG)
1170 KY$(JG) = WK$
1175 IX(JG) = W
1200 J = J - GA
1210  IF J > 0 THEN  GOTO 1130
1400 I = I + 1
1410  IF I <  = K THEN  GOTO 1120
1500 GA =  INT (GA / 2)
1510  IF GA >  = 1 THEN  GOTO 1110
1990  RETURN
```

Fig. 5. The actual sorting is done by this Shell sort routine. The routine is an implementation of the Shell sort algorithm from Kernighan and Plauger's book, Software Tools.

Sorting the File

I won't attempt to give a tutorial on sorting and sorting algorithms here. There are several excellent sources of information on these subjects (see references at the end of this article). I used the "Shell sort" algorithm from *Software Tools* by Kernighan and Plauger (p. 106) for the specification of the sorting routine for this file sort program because, although it is appreciably faster than the simple "bubble sort," it is easy to program and check out. I modified Kernighan and Plauger's algorithm to reflect the variable names I had chosen for my program (see Example 2).

The basic strategy of a Shell sort is to set an initial gap size (about half the length of the file) and compare the elements which are gap-size apart in the file, exchanging them as necessary. The gap is then reduced and the process repeated. When the gap has been reduced to zero, the sort is complete.

I used the above specification to write the sort routine that appears in Fig. 5. This routine is a direct translation of the specification into Applesoft II BASIC. I did not expect that the routine would run correctly the first time, so I included lines 1131 and 1132 to print some debugging information during the execution of the routine. I also knew that the top-level program would call the debugging routine at line 10000 after it called the sorting routine, which would show me the keys and their indices after sorting.

```
----------------
<HIGH J> <COMPAN>GA=3 I=4 J=1 JG=4
<GOLDIG> <MAGICI>GA=3 I=5 J=2 JG=5
<STRONG> <>GA=3 I=6 J=3 JG=6
<COMPAN> <GOLDIG>GA=1 I=2 J=1 JG=2
<GOLDIG> <>GA=1 I=3 J=2 JG=3
<COMPAN> <>GA=1 I=3 J=1 JG=2
<GOLDIG> <HIGH J>GA=1 I=4 J=3 JG=4
<HIGH J> <MAGICI>GA=1 I=5 J=4 JG=5
<MAGICI> <STRONG>GA=1 I=6 J=5 JG=6
AT ROUTINE 3000
0 0 WINDBAG
1 0
2 4 COMPANION
3 2 GOLDIGGER
4 1 HIGH JUMPER
5 5 MAGICIAN
6 3 STRONG ARM
```

Fig. 6. The results obtained during the first debugging run show that the sort routine and the routine which extracts the keys are in disagreement about the location of the keys.

The first debugging run for the sort routine is shown in Fig. 6. The sorting routine printed, just prior to each comparison, the j[th] and the jg[th] keys and the values of GA, I, J and JG. After the sorting routine had returned, the top-level program called the stub for the routine at line 3000 and then called the debugging routine at line 10000, which dumped the indices and keys.

```
gap = K/2
DO WHILE gap>0
    i = gap + 1
    DO WHILE i<= K
        j = i - gap
        DO WHILE j>0
            jg = j + gap
            IF j - th key>jg - th key
                THEN exchange the J - th and jg - th keys and their indices
            j = j - gap
        END DO WHILE
        i = i + 1
    END DO WHILE
    gap = gap/2
END DO WHILE
RETURN
```

Example 2.

A little study revealed that in some sense everything seemed to be off by 1. For example, the first line of debugging output says that the first and fourth keys are being compared, but referring back to Fig. 1 (part 1), I saw that "HIGH JUMPER" and "COMPANION" were really the second and fifth keys! Further study verified that although the routine which extracted the keys put the first key in KY$(0), the second key in KY$(1), etc., the sort routine was written as though the first key was in KY$(1), the second in KY$(2), etc.

I could have changed either of these routines to correct the problem. I chose to change the key extraction routine rather than the sorting routine, simply because I thought that would be the easiest to change. I made the following changes to the key extraction routine:

```
2147 KY$(K + 1) = MID$(WK$,KF,KL)
2170 KY$(K + 1) = LEFT$(WK$,KL)
2180 IX(K + 1) = K
```

Fig. 7 shows the results I got from the next debugging run. Everything seemed to be working correctly! The keys were in sort, and the indices had the correct values. For example, the results showed that after the sort the first key was "COMPANION" (it was now in KY$(1)), and that it came from the fourth record group (because IX(1) contained a 4). What I had thought would be the only hard part of the program had turned out to be quite simple! Only one routine remained to be written.

Writing the Sorted File out to the Disk

My original plan had been to

write a program that would extract the key fields and their indices from an unsorted file, sort them and use the sorted indices to read the unsorted file (in sort order) and write out a new file which would be sorted. As I approached the task of writing the routine that would read the unsorted file and write the sorted file, it occurred to me that I would rather make use of a concept known as indexed sequential files.

Under my original plan, if I needed to have a file sorted in three different ways (e.g., by last name, by city and by zip code), I would have to keep three full copies of the file. If the file was at all large, this could become a serious problem. An indexed sequential file is really two files: a data file and an index to it. The index contains the record (or record group) numbers arranged in some special order and can be used to read the file according to some special sequence.

For example, the index prepared by the sort routine can be used to read the unsorted file in the sorted sequence. The index constructed during the debugging run shown in Fig. 7 can be used to read the unsorted file in a sequence, which is sorted by occupation. Thus it says that if you want to read the file in sort order by occupation, you should first read record group 4, then record group 2, then record group 1, etc. The index file will be much smaller than the actual file, and you could afford to have several index files (for different sort orders) at a relatively small expense in disk space.

As a result of these considerations, I decided to simply write out the sort index rather than the sorted file[2]. After making this decision, I wrote the routine shown in Fig. 8. This routine simply opens the file SF$ for

[2]You might enjoy, as an interesting and useful exercise, adding a new routine that gives the user the option of writing out the sorted file. The routine at line 7000 can easily be modified to do this job.

```
7000  REM  ----------------------
7010  REM  LIST THE SORTED FILE
7020  REM  ----------------------
7100  INPUT "DO YOU WANT TO LIST THE SORTED FILE? ";AN$
7110  IF  LEFT$ (AN$,1) <  > "Y" THEN  GOTO 7990
7120  REM  TURN ON THE PRINTER
7130  GOSUB 8000
7150  REM  OPEN THE DATA FILE
7160  PRINT D$;"OPEN ";DF$;",L";RL + 1
7170  FOR JJ = 1 TO K
7180  FOR KK = 1 TO RG
7190  PRINT D$;"READ ";DF$;",R";HS + IX(JJ) * RG + KK - 1
7200  INPUT LI$
7210  PRINT LI$
7220  NEXT KK
7230  IF RG > 1 THEN  PRINT : PRINT : PRINT
7240  NEXT JJ
7250  REM  TURN OFF THE PRINTER
7260  GOSUB 9000
7990  RETURN
8000  REM  ---------------------
8010  REM   TURN ON THE PRINTER
8020  REM  ---------------------
8100  PRINT D$;"BLOAD RS232 DOS 36K"
8110  CALL 768
8120  RETURN
9000  REM  ----------------------
9010  REM   TURN OFF THE PRINTER
9020  REM  ----------------------
9100  PRINT D$;"PR#0"
9110  RETURN
```

Fig. 9. Uses the sorted index prepared by the Shell sort routine to list the unsorted file in sorted order. It uses the direct access feature of Apple DOS to read the records in the proper order.

```
----------------
<WINDBA> <STRONG>GA=3 I=4 J=1 JG=4
<HIGH J> <COMPAN>GA=3 I=5 J=2 JG=5
<GOLDIG> <MAGICI>GA=3 I=6 J=3 JG=6
<STRONG> <COMPAN>GA=1 I=2 J=1 JG=2
<STRONG> <GOLDIG>GA=1 I=3 J=2 JG=3
<COMPAN> <GOLDIG>GA=1 I=3 J=1 JG=2
<STRONG> <WINDBA>GA=1 I=4 J=3 JG=4
<WINDBA> <HIGH J>GA=1 I=5 J=4 JG=5
<STRONG> <HIGH J>GA=1 I=5 J=3 JG=4
<GOLDIG> <HIGH J>GA=1 I=5 J=2 JG=3
<WINDBA> <MAGICI>GA=1 I=6 J=5 JG=6
<STRONG> <MAGICI>GA=1 I=6 J=4 JG=5
<HIGH J> <MAGICI>GA=1 I=6 J=3 JG=4
AT ROUTINE 3000
0 0
1 4 COMPANION
2 2 GOLDIGGER
3 1 HIGH JUMPER
4 5 MAGICIAN
5 3 STRONG ARM
6 0 WINDBAG
```

Fig. 7. The corrections made to the key extraction routine clear up the problems. The Shell sort routine is working correctly.

```
3000  REM  ----------------------
3010  REM   WRITE THE SORT INDEX
3020  REM         FILE OUT
3030  REM  ----------------------
3100  PRINT D$;"OPEN ";SF$
3110  PRINT D$;"WRITE ";SF$
3120  FOR I = 1 TO K
3130  PRINT IX(I)
3140  NEXT I
3150  PRINT "END-OF-FILE"
3160  PRINT D$
3170  RETURN
```

Fig. 8. This routine simply opens the file for writing, writes out the sorted index, writes an end-of-file signal and turns off the writing of the file.

writing, writes the index to this file and writes "END-OF-FILE" as an end-of-file signal. The only interesting aspect of this routine is the statement at line 3160.

As I discussed previously, the OPEN and WRITE commands to Apple DOS say, "until I tell you otherwise, all PRINT statements should write data to the specified file." There are two ways of rescinding this command. The first is to issue a CLOSE command (see Fig. 7, part 1). The second way to turn off the writing is to issue *any other* Apple DOS command. Apple DOS commands are issued by executing a print statement that begins a line with a control-D. Therefore, line 3160 will issue an Apple DOS command when it is executed.

It is a peculiar Apple DOS command because it *only* contains a control-D. Such a statement issues a do-nothing command to Apple DOS, but since *any* command to Apple DOS shuts off any currently active WRITE (or READ) commands, it does so for the WRITE command issued at line 3110. Line 3160 is only included as a precautionary measure. When that statement is executed, the routine has finished writing the index out to the file. It ensures that any subsequent PRINT statements won't cause data to be written to the index file.

I could have simply closed the file, but for no particularly important reason I had chosen to close all files in the wrap-up routine at line 4000. The statement at line 3160 would protect the index file from subsequent PRINT statements and allow other routines to be executed prior to the closing of the file in the wrap-up routine.

In order to make sure that I did not forget to close the index file, I added the statement:

```
4100 PRINT D$;"CLOSE";SF$
```

to the routine at 4000 (see Fig. 7, part 1).

Having decided to write out just the sort index, rather than a sorted file, I also decided that I should add a routine that would allow the user to (optionally) have the sorted file printed or displayed. Since the routine at line 3000 was quite simple, I decided to write the printing routine and use it to check out the routine which writes out the index.

Printing the Sorted File

My first step in adding the printing routine was to add the following statements to the

top-level program:

```
580 LIST THE SORTED FILE
590 GOSUB 7000
600 END
```

I then wrote the routine shown in Fig. 9. This routine is similar to the extract keys routine. There are, however, a couple of interesting features that I should point out.

Perhaps the routine's most important characteristic is that it assumes that the user has a printer on the system. The routine at line 8000 is the one that is supposed to turn on the printer, and the routine at line 9000 turns it off. If you do not have a printer, the lines:

```
8025 RETURN
9025 RETURN
```

should be added to the program. Adding those two lines will deactivate those routines, and the routine at line 7000 will simply cause the sorted file to be displayed on the television or monitor screen.

Although lines 8100 and 8110 are peculiar to my particular printer and the software driver which it requires, I have included those lines in order to illustrate another aspect of Apple DOS. Line 8100 issues an Apple DOS command. It has the same effect that typing BLOAD RS-232 DOS 36K on the keyboard does. Thus, the command issued by line 8100 causes the binary file named RS-232 DOS 36K to be loaded from the disk. This is the name of the software driver which is required for my particular printer. The CALL statement at line 8110 causes the printer to be activated by calling the software driver loaded by the previous statement.

The routine that extracted the key fields (see Fig. 1) used the direct access feature to read only the first record and the key-containing record within each record group. The routine at line 7000 uses the same feature to read record groups from the unsorted file in sorted order. It uses the index file (which need not be read from the file since it is still in the array IX) to decide which record group should be read next.

You should study lines 7170 through 7220 to see how this is done. The FOR JJ loop uses the index IX(JJ) to determine which record group should be read next. The FOR KK loop then reads and prints the records in the selected record group. If our previous discussions have been clear, you should have no difficulty understanding how this routine does its job.

Fig. 10a shows the results I obtained when I sorted the file shown in Fig. 1a (part 1) using characters 1–12 of the third record as the sort key and had it printed by the routine at line 7000. Fig. 10b shows the results of sorting the same file on a key defined as all 20 characters of the first record of each record group.

An Application of the File Sort Program

An interesting application of the file sort program is its use to produce a special kind of directory of magazine articles. This kind of directory was originally called a KWIC Index. KWIC stands for Key-Word-In-Context. A KWIC index is built by taking (in this case) magazine titles and rotating the words in the title so that we get one line for each important word in the title. Then sort the resulting lines.

The result is an alphabetic list that contains each title many times. In fact, each title appears once in the KWIC index for each important word in

a.
```
BULLWINKLE GEN.
NORTH WOODS
COMPANION    CLASS 2A

CINDERELLA
SHANTYTOWN
GOLDIGGER    CLASS 4A

NIMBLE JACK B.
CANDLESTICK PARK
HIGH JUMPER CLASS 1B

CROWLEY ALEISTER
HIGHER PLANES
MAGICIAN     CLASS 1A

KENT CLARK
METROPOLIS
STRONG ARM   CLASS 3A

BLOW JOE
CHICAGO
WINDBAG      CLASS 1A
```

b.
```
BLOW JOE
CHICAGO
WINDBAG      CLASS 1A

BULLWINKLE GEN.
NORTH WOODS
COMPANION    CLASS 2A

CINDERELLA
SHANTYTOWN
GOLDIGGER    CLASS 4A

CROWLEY ALEISTER
HIGHER PLANES
MAGICIAN     CLASS 1A

KENT CLARK
METROPOLIS
STRONG ARM   CLASS 3A

NIMBLE JACK B.
CANDLESTICK PARK
HIGH JUMPER CLASS 1B
```

Fig. 10. Results of sorting the file on the first 12 characters of the third record and the contents of the first record are shown. Sorted by occupation (a) and sorted by name (b).

a.
```
BUSINESS APPLICATIONS A SYSTEM FOR THE PROFESSIONAL -BREHM JA78
GROWING WITH KIM EXPANSION PC BOARD -EATON JA78
HAS GODBOUT DONE IT AGAIN? ECONORAM II -GARETZ JA78
THE TRS-80 HOW DOES IT STACK UP? -JUGE JA78
A TALE OF TWO BASICS WHICH ONE FOR YOUR 6800 -DIDDAY JA78
```

b.
```
BUSINESS APPLICATIONS A SYSTEM FOR THE PROFESSIONAL -BREHM >< JA78
APPLICATIONS A SYSTEM FOR THE PROFESSIONAL -BREHM >< BUSINESS JA78
SYSTEM FOR THE PROFESSIONAL -BREHM >< BUSINESS APPLICATIONS A JA78
PROFESSIONAL -BREHM >< BUSINESS APPLICATIONS A SYSTEM FOR THE JA78
-BREHM >< BUSINESS APPLICATIONS A SYSTEM FOR THE PROFESSIONAL JA78
GROWING WITH KIM EXPANSION PC BOARD -EATON >< JA78
KIM EXPANSION PC BOARD -EATON >< GROWING WITH JA78
EXPANSION PC BOARD -EATON >< GROWING WITH KIM JA78
PC BOARD -EATON >< GROWING WITH KIM EXPANSION JA78
BOARD -EATON >< GROWING WITH KIM EXPANSION PC JA78
-EATON >< GROWING WITH KIM EXPANSION PC BOARD JA78
HAS GODBOUT DONE IT AGAIN? ECONORAM II -GARETZ >< JA78
GODBOUT DONE IT AGAIN? ECONORAM II -GARETZ >< HAS JA78
DONE IT AGAIN? ECONORAM II -GARETZ >< HAS GODBOUT JA78
AGAIN? ECONORAM II -GARETZ >< HAS GODBOUT DONE IT JA78
ECONORAM II -GARETZ >< HAS GODBOUT DONE IT AGAIN? JA78
II -GARETZ >< HAS GODBOUT DONE IT AGAIN? ECONORAM JA78
-GARETZ >< HAS GODBOUT DONE IT AGAIN? ECONORAM II JA78
TRS-80 HOW DOES IT STACK UP? -JUGE >< THE JA78
HOW DOES IT STACK UP? -JUGE >< THE TRS-80 JA78
DOES IT STACK UP? -JUGE >< THE TRS-80 HOW JA78
STACK UP? -JUGE >< THE TRS-80 HOW DOES IT JA78
UP? -JUGE >< THE TRS-80 HOW DOES IT STACK JA78
-JUGE >< THE TRS-80 HOW DOES IT STACK UP? JA78
TALE OF TWO BASICS WHICH ONE FOR YOUR 6800 -DIDDAY >< A JA78
TWO BASICS WHICH ONE FOR YOUR 6800 -DIDDAY >< A TALE OF JA78
BASICS WHICH ONE FOR YOUR 6800 -DIDDAY >< A TALE OF TWO JA78
WHICH ONE FOR YOUR 6800 -DIDDAY >< A TALE OF TWO BASICS JA78
ONE FOR YOUR 6800 -DIDDAY >< A TALE OF TWO BASICS WHICH JA78
6800 -DIDDAY >< A TALE OF TWO BASICS WHICH ONE FOR YOUR JA78
-DIDDAY >< A TALE OF TWO BASICS WHICH ONE FOR YOUR 6800 JA78
```

Fig. 11. Collection of magazine titles rotated and replicated in preparation for sorting. The sorted file will provide a KWIC index to the titles. (a) Partial sample of titles from the January 1978 issue of Kilobaud. *(b) Titles rotated and replicated so that there is one copy for each important word in the title.*

the title. Such an index is useful when you only remember the general subject or a word or two from the title of the article you are looking for. I have a program that, given a file containing the titles, prepares a new file containing the rotated titles (which can then be sorted by the file sort program to produce a KWIC index). In Fig. 11a I have shown a small sample file which contains some of the article titles from the January 1978 issue of *Kilobaud*. The file, which was produced by the program that rotates the titles, is shown in Fig. 11b. The rotating program doesn't know what important words are; it just contains a list of unimportant words such as a, the, etc. Finally, Fig. 12 shows the KWIC index which was produced by the file sort program by sorting the file shown in Fig. 11b.

```
-BREHM >< BUSINESS APPLICATIONS A SYSTEM FOR THE PROFESSIONAL JA78
-DIDDAY >< A TALE OF TWO BASICS WHICH ONE FOR YOUR 6800 JA78
-EATON >< GROWING WITH KIM EXPANSION PC BOARD JA78
-GARETZ >< HAS GODBOUT DONE IT AGAIN? ECONORAM II JA78
-JUGE >< THE TRS-80 HOW DOES IT STACK UP? JA78
6800 -DIDDAY >< A TALE OF TWO BASICS WHICH ONE FOR YOUR JA78
AGAIN? ECONORAM II -GARETZ >< HAS GODBOUT DONE IT JA78
APPLICATIONS A SYSTEM FOR THE PROFESSIONAL -BREHM >< BUSINESS JA78
BASICS WHICH ONE FOR YOUR 6800 -DIDDAY >< A TALE OF TWO JA78
BOARD -EATON >< GROWING WITH KIM EXPANSION PC JA78
BUSINESS APPLICATIONS A SYSTEM FOR THE PROFESSIONAL -BREHM >< JA78
DOES IT STACK UP? -JUGE >< THE TRS-80 HOW JA78
DONE IT AGAIN? ECONORAM II -GARETZ >< HAS GODBOUT JA78
ECONORAM II -GARETZ >< HAS GODBOUT DONE IT AGAIN? JA78
EXPANSION PC BOARD -EATON >< GROWING WITH KIM JA78
GODBOUT DONE IT AGAIN? ECONORAM II -GARETZ >< HAS JA78
GROWING WITH KIM EXPANSION PC BOARD -EATON >< JA78
HAS GODBOUT DONE IT AGAIN? ECONORAM II -GARETZ >< JA78
HOW DOES IT STACK UP? -JUGE >< THE TRS-80 JA78
II -GARETZ >< HAS GODBOUT DONE IT AGAIN? ECONORAM JA78
KIM EXPANSION PC BOARD -EATON >< GROWING WITH JA78
ONE FOR YOUR 6800 -DIDDAY >< A TALE OF TWO BASICS WHICH JA78
PC BOARD -EATON >< GROWING WITH KIM EXPANSION JA78
PROFESSIONAL -BREHM >< BUSINESS APPLICATIONS A SYSTEM FOR THE JA78
STACK UP? -JUGE >< THE TRS-80 HOW DOES IT JA78
SYSTEM FOR THE PROFESSIONAL -BREHM >< BUSINESS APPLICATIONS A JA78
TALE OF TWO BASICS WHICH ONE FOR YOUR 6800 -DIDDAY >< A JA78
TRS-80 HOW DOES IT STACK UP? -JUGE >< THE JA78
TWO BASICS WHICH ONE FOR YOUR 6800 -DIDDAY >< A TALE OF JA78
UP? -JUGE >< THE TRS-80 HOW DOES IT STACK JA78
WHICH ONE FOR YOUR 6800 -DIDDAY >< A TALE OF TWO BASICS JA78
```

Fig. 12. A KWIC index of a small sample of titles shows the utility of such an index. The use of the hyphen preceding the author's name produces an index that begins with an author index of the titles. The remainder of the index makes it easy to locate an article even when you know only the general subject or one or two words in the title.

I hope you have learned something which will be useful to you as you have read this discussion, that you may find the file sort program useful and that you enjoy your Apple II as much as I do. ■

References

Kernighan and Plauger, *Software Tools.*

D. E. Knuth, *The Art of Computer Programming*, Vol. 1.

R. Sedgewick, *Implementing Quicksort Programs*, Comm. of the ACM, October 1978.

Gregory Harris
5334 Oak
Kansas City, MO 64112

Elf Meets a New Friend

In this case, the Elf's pal is an SWTP 6800 memory board.

Recently I was in the same dilemma as other computer hobbyists—I needed more money! I had wire-wrapped a Cosmac Elf with 256 bytes of RAM according to an article in *Popular Electronics*, but it was begging for more memory.

As a part-time hobbyist, I have limited time and a limited budget. At the time, few companies offered bare PC boards, which I considered the best purchase. Bare boards are usually inexpensive; in the case of a memory board, I can purchase a 1K or 2K segment at a time as my budget allows.

I had had previous experience with SWTP in the audio field, and I was pleased with the price and quality of their products. At the time I placed my order with SWTP, only a 4K bare board was available. My next move was obvious: to mate my 1802-based Elf with the SWTP 6800 memory board.

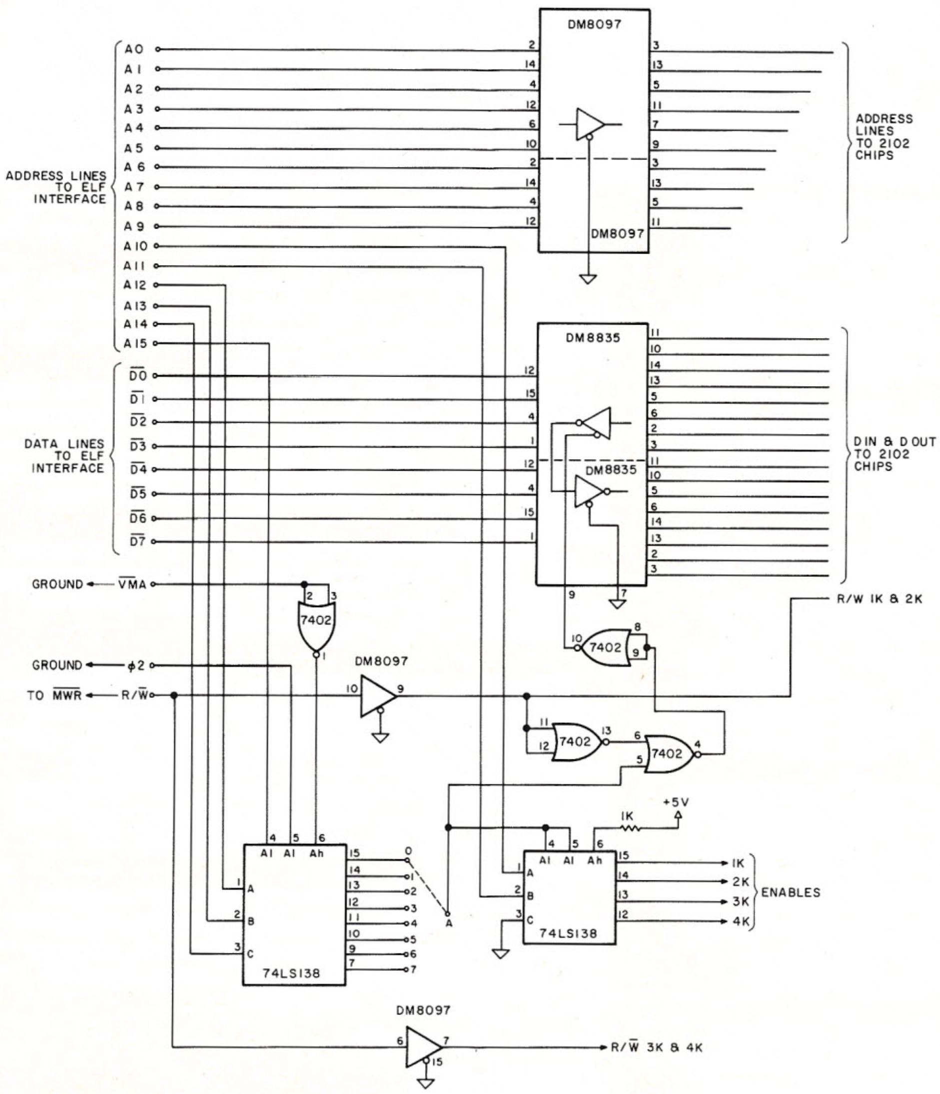


Fig. 1. Buffering and decoding section of the SWTP 4K memory board. Elf interface connections shown on left.

4K Board Description

SWTP's board design is based on 2102 memory chips. Buffering and decoding are done with reliable TTL circuitry. As you can see from Fig. 1, there are 16 memory address lines, ten of which (A_0 through A_9) are buffered with DM8097 Tri-state non-inverting hex buffers. The other six lines, A_{10} through A_{15}, are routed to the 74LS138 3 to 8 decoders for address decoding. This allows placement of the board in any 4K segment of the total computer memory and also allows decoding of the individual 1K segments on each board.

The eight bidirectional data lines are buffered and controlled by DM8835 inverting bidirectional transceivers. Although the incoming sections are enabled at all times, this doesn't cause any problems because the 2102s have separate data in and data out lines. The output sections of the DM8835 chips are controlled by the R/$\overline{W}$ line and signals from the address decoder that are routed through a series of gates

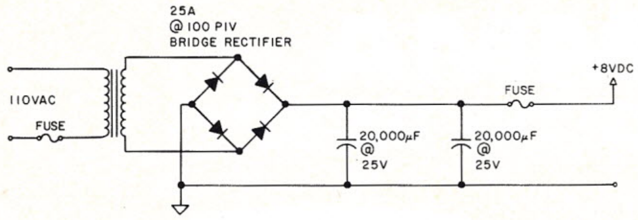


Fig. 2. Expansion power supply. Fuses selected according to load.

in the form of a 7402 quad 2 input NOR gate. The R/$\overline{\text{W}}$ line is previously buffered by a part of the DM8097 address line buffer.

The only other signal lines are $\overline{\text{VMA}}$ and Ø2, which are used to enable the first address decoder. $\overline{\text{VMA}}$ is valid low and is inverted by a section of the 7402. The only two lines remaining are the unregulated V+ line and ground. The V+ line supplies the on-board 7805 regulators to supply +5 V dc to all of the chips.

Putting It All Together

I searched through all of my discount-house catalogs to find the best prices for chips and sockets. By careful shopping I was able to keep the cost below $80, which at the time was reasonable for 4K of static RAM. Assembly was easy. I soldered all the sockets, capacitors and resistors in place and then simply installed the chips.

A friend with an SWTP 6800 computer let me install the board in his mainframe to check the operation of the memory. It worked flawlessly! My next step was to interface the board to my 1802. The Elf needed a heftier power supply, and the address, data and signal lines needed buffering.

Elf Additions

The power supply of the original Elf supplied only approximately 600 mA. Any major memory additions required power in Amperes instead of milliamps, so I chose to build a supply that could deliver 6 A to accommodate any future expansion. I purchased a surplus transformer, and as you can see from Fig. 2, I used a standard circuit to supply approximately 8 volts dc to the system.

My next concern was the address lines. RCA uses multiplexing to place 16 bits of information on 8 pins. The high-order 8 bits of the address are placed on the pins during the first half of a machine cycle, and the low-order 8 bits are placed on the same pins during the remainder of a machine cycle.

The 1802 generates a timing pulse, TPA, during the first part of a cycle that can be used to strobe the high-order bits into a latch. After TPA goes low, the low-order address bits are present. I used two CMOS CD4042 quad latches to capture the high-order bits for address lines A_8 through A_{15} (see Fig. 3). After this, I buffered the lines with DM8097 hex buffers and used the extra sections to buffer the $\overline{\text{MRD}}$ and the $\overline{\text{MWR}}$ lines.

For the data lines I used the same transceivers as on the memory board to keep things simple. I used the $\overline{\text{MRD}}$ and $\overline{\text{MWR}}$ lines to gate the transceivers in and out, respectively. Since the signals ($\overline{\text{MRD}}$ and $\overline{\text{MWR}}$) are never active at the same time, the data direction cannot be confused. Fig. 4 shows the simple circuit.

The 1802 has built-in provisions for DMA by way of two lines, $\overline{\text{DMA-in}}$ and $\overline{\text{DMA-out}}$. When either of these two lines is brought low, the CPU prepares for DMA and gives up control of the data lines by placing them in a high-impedance state. For this reason, I didn't have to decode a high-impedance state for the data buffers to isolate the CPU data lines from the bus, as SWTP does for the 6800 computer.

The last two signals were $\overline{\text{VMA}}$ and Ø2. Fig. 1 shows that these signals tell the address decoder that the 6800 CPU is outputting a valid memory address. To keep my parts count to a minimum, I tried grounding these two inputs to the memory board instead of synthesizing a signal to emulate the 6800. The memory operated without an error.

I experimented with the $\overline{\text{VMA}}$ input by putting TPA on it to keep the address decoder on the memory board from responding until after the high-order memory address bits were latched and the low-order bits were present. The board also operated OK in this mode.

Interface Wiring

I wired between the Elf and the memory board with two DIP connectors and ribbon cable, keeping the cable as short as possible to avoid any cross-talk problems. I wire-wrapped the buffering and decoding logic on a piece of perfboard the same size as the Elf to allow room for expansion. Everything on my expansion interface board was mounted in sockets for easy assembly and troubleshooting.

The result of my experiment put me one step closer to my goal of running Tiny BASIC and also provided me with a lot of fun and insight into the operation of the 1802 and also the 6800 CPUs. I plan to add a keyboard and video monitor, as well as I/O ports to accommodate my future projects. ■

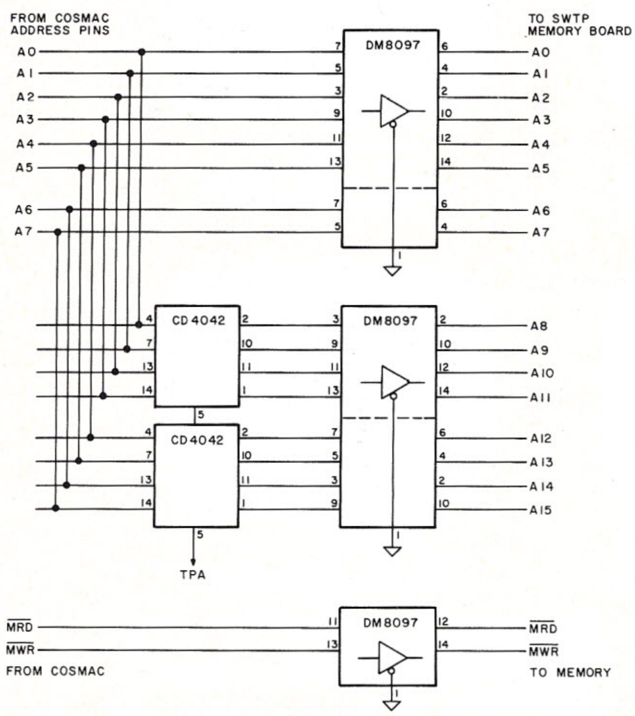


Fig. 3. Method used to latch and buffer the address lines. Also, buffering for the $\overline{\text{MRD}}$ and $\overline{\text{MWR}}$ lines.

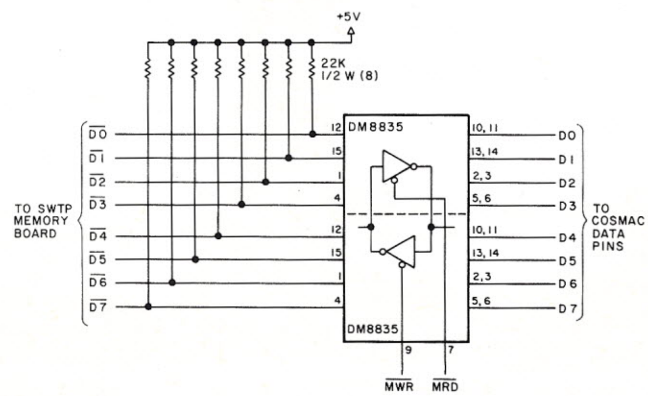
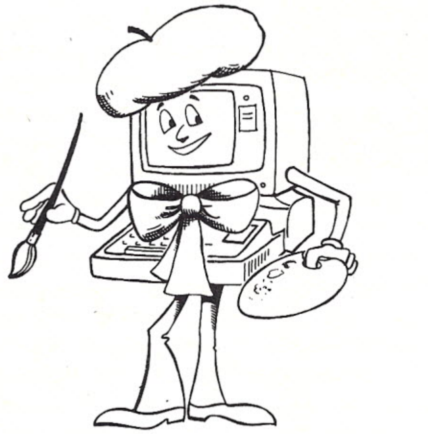


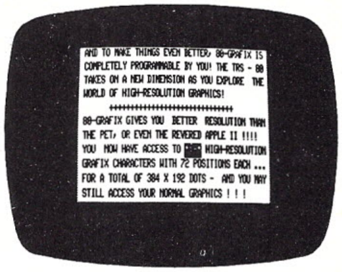
Fig. 4. Expansion interface data line buffers. Also note bus pull-up resistors.

FROM **PROGRAMMA**

HI-RESOLUTION GRAPHICS FOR THE TRS-80®

LOWER CASE
The 80-GRAFIX board includes two sets of lower case characters at no additional cost.





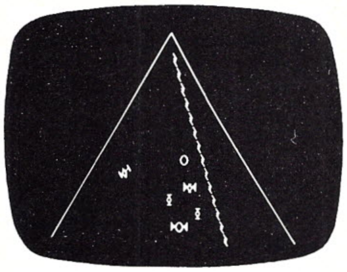
INVERSE VIDEO
The 80-GRAFIX board allows you to do inverse video to high-light your screen displays.

FINALLY, AT LAST. . .

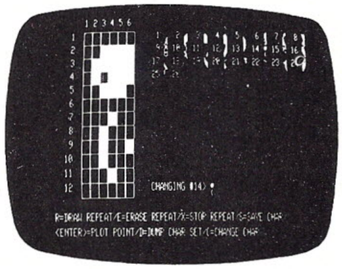
HI- RESOLUTION GRAPHICS is available for your TRS-80 computer system. The 80-GRAFIX board from PROGRAMMA International, Inc. gives your TRS-80 high resolution capability that is greater than the Commodore CBM/PET or even the revered APPLE II.

80-GRAFIX gives the TRS-80 an effective screen of 384X192 pixels, versus the normal 127X192 for the TRS-80, 80X50 for the CBM/PET, or the 280X192 of an APPLE II. As an added feature, 80-GRAFIX offers you lower case characters at no additional cost. Of course, you can also create your own set of up to 64 original characters using the supplied Character Generator software.

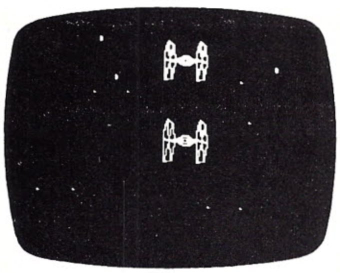
The 80-GRAFIX board is simple to install (note that this voids your Radio Shack warranty), and programming is done through BASIC. 80-GRAFIX opens up a whole new realm of software development and excitement never dreamed of for the TRS-80!



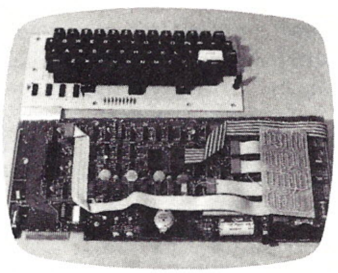
DEMONSTRATION PROGRAMS
The 80-GRAFIX board is supplied with a Character Generator software and several demonstration programs.



CHARACTER GENERATOR
The supplied character generator software allows you to create your own character set of up to 64 original characters.

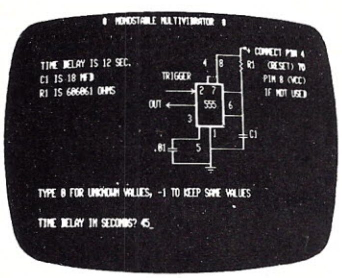


REAL-TIME GRAPHIC GAMES
With the 80-GRAFIX board you can write exciting real-time games using BASIC.

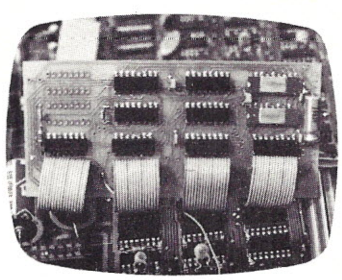


EASY INSTALLATION
The 80-GRAFIX board is simple to install and fits inside the TRS-80 case.

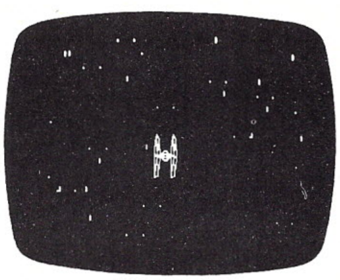
GRAPHICS GALORE
The 80-GRAFIX board and the supplied Character Generator allow you to become an artist.



ELECTRONIC DESIGN
The 80-GRAFIX board has unlimited application in Electronic design and Education.



80-GRAFIX HI-RESOLUTION
Finally, the only means to protect your computer investment is to order an 80-GRAFIX board TODAY!



EXCITEMENT & FUN
Open up a new realm of software development with the 80-GRAFIX board.

Available exclusively through PROGRAMMA at the cost of $149.95
Please check with us for availability prior to ordering
VISA and MASTERCHARGE accepted
TRS-80 is a registered trademark of the Tandy Corp.

PROGRAMMA INTERNATIONAL, INC.
3400 Wilshire Blvd.
Los Angeles, CA 90010
(213) 384-0579 • 384-1116 • 384-1117

✓11

TRS 80

MODEL 3
call today!

TRS-80 Model I

699

Level II 16K. 26-1056

Model II

$3499

64K, 1-Disk TRS-80
Model II System

We accept check, money order or phone orders with Visa or Master Charge. (Shipping costs added to charge orders).

master charge VISA CHARGE IT

Computers Unlimited

1524 OAK HARBOR ROAD, FREMONT, OHIO 43420

419-332-4881 Collect

26

TRS-80 is a Trademark of Tandy Corp.

STOCK CONTINUOUS CHECKS & STATEMENTS

FREIGHT INCLUDED

We carry forms specifically designed for these software formats:

- TRS-80 (Radio Shack)*
- Durango
- TCS
- Peachtree**

SOON TO BE ADDED—Checks for Osborne packages

*Low minimum orders *Fast delivery *Quantity discounts

SEND FOR YOUR FREE SAMPLES TODAY!

The Best Choice for the Small Computer User

* Personal Checks
* Payables Checks
* Payroll Checks
* Combination Checks
* Statements
* Invoices

NATIONWIDE SERVICE

Checks To-Go 58

* Registered Trademark of Tandy Corp.

**Registered Trademark of Retail Sciences, Inc.

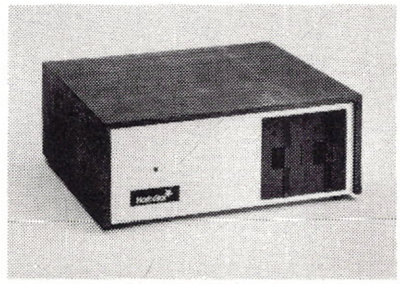
8384 Hercules St., La Mesa, CA 92041, (714) 460-4975

Save more than 20%!

AMERICAN SQUARE COMPUTERS

**NORTH STAR—INTERTUBE—MICROTEK
ZENITH—HEATH—ITHACA
THINKER TOYS—GODBOUT—SOFTWARE**

The smartest computers at the smartest price.



FACTORY ASSEMBLED & TESTED	LIST	ONLY
HORIZON-1-16K-DOUBLE DEN KIT	SPECIAL	$1269
HORIZON-1-32K-DOUBLE DEN KIT	$1999	1575
HORIZON-2-32K-DOUBLE DEN KIT	2399	1879
HORIZON-1-32K-DOUBLE DEN	2695	2129
HORIZON-2-32K-DOUBLE DEN	3095	2435
HORIZON-2-32K-QUAD DENSITY	3595	2839
HORIZON-2-64K-QUAD + HARD DISK	9329	7229
HORIZON MEMORY 16K 389 32K 579		
NORTH STAR HARD DISK 18 Mb	4999	3949
PASCAL FOR NORTH STAR ON DISK	199	190
Powerful NORTH STAR BASIC.......... The Best..........		FREE
2 NORTH STAR SOFTWARE DISKS w/HORIZON..........		FREE
NORTH STAR BUSINESS PROGRAMS & NORTHWORD,		PHONE
COLOR! RAINBOW-2000 & CAT-100		PHONE
ITHACA FRONT PANEL COMPUTER 64K	2885	2449
Z-8000 CPU CARD 16-bit ITHACA S-100		PHONE
ITHACA MEMORY 8/16-bit		PHONE
8086 CPU 16 bit 10 × faster SEATTLE COMPUTER		
SEATTLE COMPUTER MEMORY		PHONE
SSM Z-80 CPU, VIDEO BOARD, MEMORY		PHONE
MEASUREMENT MEMORY 64K A & T 4mHz		650
JAWS MEMORY 64K A & T 4mHz		PHONE
GODBOUT MEMORY—Static, Super Selection & Price		
THINKER TOYS DISCUS/2D A & T	1199	975
THINKER TOYS HARD DISK 26 Mb	4995	4149
DISCUS/2 + 2 1.2 Mbytes A & T	1545	1285
THINKER TOYS SUPERRAM		PHONE
DELTA COMPUTER & DISK DRIVES		PHONE
TARBELL COMPUTERS & DISK DRIVES		PHONE
INTERTUBE II SMART TERMINAL	995	725
ZENITH-HEATH SMART TERMINAL Z-19 A & T		795

ZENITH COMPUTER-TERMINAL-DISK Z-89	2595	2195
CAT NOVATION MODEM	179	169
MICROTEK PRINTER	795	725
AXIOM PRINTER	795	695
ANADEX PRINTER	995	865
NEC PRINTER Fast Typewriter Quality	2915	2799
SECRETARY WORD PROCESSOR The Best!	85	77
TEXTWRITER III Book Writing Program	125	112
GOFAST NORTH STAR BASIC Speeder Upper	79	71
PDS Super Z-80 ASSEMBLER & More	99	89
COMPILER FOR NORTH STAR $150 w/PDS & HDS		90
EZ-80 MACHINE LANGUAGE TUTORIAL $25 HDS		40
EZ-CODER Translates English to BASIC	79	71
ECOSOFT FULL ACCOUNTING PKG	350	315
THE SOURCE $90.00 DATABASE CROSS ASSEMBLERS		CALL
BOX OF DISKETTES 29 IN PLASTIC CASE 30		
Which Computers are BEST? BROCHURE..................		FREE
North Star Documentation refundable w/HRZ		20
ORDER 2 or more COMPUTERS..........		BIGGER DISCOUNTS

FACTORY ASSEMBLED & FACTORY WARRANTY

AMERICAN SQUARE COMPUTERS 56

KIVETT DR
JAMESTOWN NC
27282
(919) 889-4577

The Future Office: In the Home

"I won't be at the office today—I've got some work to do at home."

Jeff Noynaert
Managing Editor
Future Office Abstracts
PO Box 29251
San Antonio, TX 78229

Developments in communication and computer technology are leading us to more versatile and distributed systems. Soon, inexpensive home-based computers will be able to simulate the functions of large office staffs.

This article describes the uses that Mr. and Ms. Doe will find for these systems and several effects that these changes will have on society. This is not seen as a direct replacement for today's methods, but as a new competitor (and tool) for them in the marketplace. The article calls for authors to develop and expand these and related subjects and to alert the American public to their future resources.

Background

Who would have thought that one day consumers would be able to perform the very functions that for so long have been the businessman's role in society? Not that business is on the way out . . . far from it. But soon the face of business will be quite different from what it is today, and the once distinct division between the roles of buyer and seller will become blurred.

When computers were first developed they were of no commercial value. After working out a few bugs, investing a few million dollars and discovering a few technological breakthroughs, several users could join together to take advantage of one system and share the expenses. Several million dollars later, the largest corporations could afford to own (or lease) one for themselves. The difference was that the cost had come down, thus making computers profitable to own. As their price continued to drop, more companies found it practical to use them.

> "What big businesses are dreaming of doing now, everyone will be able to do in only a few more years."

This is similar to the way that word processing was introduced into business offices. First, word processing was so costly that its only practical use was correcting computer programs . . . and only then because there was a computer involved and waiting at the time. With advances in circuit integration and available research and development dollars, a true word processor was offered to the business community.

Still, the best ones were more expensive than the secretaries that they replaced, and users had to settle for paper tape machines, if they could justify having one at all. With time, word processors began using magnetic media (because magnetic media had become less expensive) and became more applicable to routine office correspondence.

Establishing word-processing centers was suggested by the vendors in order to serve several persons and therefore distribute the cost . . . much like time-sharing one computer between several businesses. Today, word processors no longer need to be shared in order to justify their expense, because today's lower prices make it possible to have them distributed throughout the company.

The point is that the cost of computers—and all the related hardware—is continuing to drop and will continue to drop for enough years to allow everyone in the country to own complex computing systems, much the same way we now own calculators and television sets.

What big businesses are dreaming of doing now, everyone will be able to do in only a few more years. Electronic mail, advanced information management and data base systems are just beginning to be implemented in the largest corporations. Vendors

throughout the world are falling over themselves to develop less expensive, simpler systems.

As always, functions are first added to a large centralized computer, making these new functions available to several users throughout the company or among several companies. Centralization always occurs first because it allows several persons to share system costs.

Eventually though, when the prices drop, the advantages of decentralization (or distributed systems) become evident. The distributed systems don't require a connection to a central host; they allow local control, reduce bottlenecks, eliminate the possibility that all the systems will go down at once, add privacy and let you personalize the features of your system. The most advanced systems involve an ever-changing balance of centralization and decentralization.

Computers were first used to automate clerical tasks. Then many other cause-and-effect jobs were described in the form of a program. Today, we have automated banks, automated order and bill processing, automated mail delivery. Anything that can be described in a certain cause-and-effect series of events will be automatic.

A Forecast

Within ten years nearly everyone will be able to afford a computer that can perform all of these business functions. Each family will be able to simulate a large office staff. Some businesses will be able to offer reduced prices by depending more on the consumers' new abilities.

Imagine this: You don't need salesmen or advertising personnel because the customer's computer can find you without them. You don't need a high-traffic location for your store because your customers aren't depending on signs for their information. You don't need people to take the orders because the customers will tie in to your computer and fill in your computerized order blank. What you need is a product and about $5000 worth of computer gear.

Before this idealistic new system reaches fruition we will go through the same stages of centralization and sharing of the necessary hardware. Only time will tell exactly how it will take place. We could install vending machine terminals in the supermarkets. Some arcade games now cost more than this setup would. On the other hand, there will also be neighbors and retailers who will offer to search the world for you with their systems.

Retailers today have overheads that require marking their wares up from 100 percent to 200 percent. The retail industry standard—from clothes to hard goods—is the keystone (100 percent) markup. Now, I'm not saying that there is anything wrong with that, or with the fact that jewelry is marked up even more. I'm saying that automation applied to merchandising will allow on-line merchants (OLMs) to sell you a $40 coat for $22. That is fair competition, too. When you have your own system, you'll do even better than that.

So decreases in cost and trends toward distributed computing will inevitably lead the whole computer industry to a major thrust into the largest market of all, the personal market. When you think in terms of large organizations, you find varied needs that split the discussion into many different directions. One hun-

dred million individuals, though, can be discussed as one group with similar needs and wants.

Mr. and Ms. Doe

As yet, we don't know what Mr. and Ms. Doe will require from computers, communications and their peripherals. In a free competitive market, though, they will get whatever they want. With millions of dollars at stake and with our foot in the door, it's about time we found those wants for them.

Let's examine what automated functions the consumer can use.

- On-line publishers and news services.
- Active idea exchanges (usually for profit).
- Lots of simplicity and standards.

No new technology is needed to develop these systems, and some of them are in use today. There is literally more technology than we know what to do with. Simultaneously, there are approaching several dramatic advances associated with communication and information management:

- Very large scale integration (VLSI) that can put minicomputers in your pocket and word processors on a chip.

> "There is literally more technology than we know what to do with."

- Electronic mail to save money, speed interaction and simplify communications.
- Automatic searches of the computers in the local area (local for low communication and transportation costs) for an idea or a product at the best quality/price ratio.
- Electronic bulletin board available to other individuals and their computers to make ideas, products and services known.
- On-line merchandising (OLM) with compiled product data bases.
- On-line original manufacturers (OLOM) for the lowest prices.
- A better business bureau, standards and trade and user groups to protect the individuals and give the system credibility.
- Distribution libraries (borrow and lend exchanges) to locate a lender of a certain thing needed only for a short time.
- Classes and round-robin discussions to stimulate your imagination and keep you up to date.
- Automated ordering of products and automated updates until those products arrive.
- Electronic funds transfer (EFT) to simplify marketing.
- Organized barter to save taxes and stimulate trade.
- Time-share services for individuals and families to balance the distributed/centralized issue and to quickly implement new features.
- Private mail (locally coded and decoded).
- Large communications networks (XTEN, ACS, SBS) to offer satellite communications, video conferencing, voice-store-and-forward and an integrated voice, data, image and text network.
- New technologies to multiply memory densities annually.
- Advanced software to develop plain-English programming and system simplicity.

Effects

We should consider some of the effects that advanced communication and automation will have on our society.

- Reduction in automobile use and possibly in cross-country distribution.
- Increase in cottage industry and self-employment.
- Personnel reductions in white-collar departments.
- Decreased retail prices.
- Middleman and retailer reductions.
- Large investments in microcomputers.

Fewer middlemen, retailers and administrators will mean more persons involved in production and direct services. This means an increased national production potential and a better balance of trade. It should also mean a higher average standard of living.

Further by-products of this advanced society include:

- The practicality of temporary, ad hoc groups organized for profit.
- Increased rate of change.
- Increased need for the public to update their education.
- Retail terminals.
- Increased interpersonal communication.
- Society will be functionally ahead of any society that uses older methods.

We should also consider how these advances will affect:

- The government (reassessment by the public, participatory democracy).
- Our culture.
- Large computer corporations.
- Copyright control.
- The third world.
- Advertising.
- Taxes and enforcement effectiveness.
- Entertainment.
- Time.

We are now facing an age of access and potential. Like every frontier that has come before, there will be many different positions to be filled and many occupations to benefit. The first to enter will probably face the most problems and the most profits and will eventually develop the most effectiveness.

Nothing is absolute; we will always have stores, sales managers, administrators and large corporations, but in the future we will have fewer of them because there will be incentives to consider other alternatives.

Conclusion

More than new technologies, we need thought and communication about our uses for today's technologies from leaders, businessmen and authors. In America, changes are made not by governmental action or mandate, but by individual choice and market support.

Most likely, the software for many of the systems described here is already available, or will be printed in magazines for public use. We need the involvement of individuals and the development of organizations (cooperation, standards and norms, establishment of values).

The bottom line is opportunity—opportunity to offer these new systems and opportunity to use them. Where there is a better way, the market will pay someone to supply it. That is one thing about our society that computers won't change.

Over the next ten years these systems will become more evident to our culture, our society, our government and Mr. and Ms. Doe. To the people involved it is here now. ■

Open Up the SWTP 6800

A bit of minor surgery maximizes the capabilities of this popular system.

Dr. Gordon W. Wolfe
Physics Department
University of Mississippi
Oxford, MS 38677

The SWTP 6800 microcomputer system is a versatile system, particularly well-suited for real-time data acquisition, analysis and control. However, some minor engineering shortcuts have left the machine with less than its full availability. With the addition of only two common integrated circuits and a few wires, you can maximize the capabilities of the computer and its bus.

Add a User Scratchpad

I recently developed a low-cost vectored graphics display and had just finished testing both software and hardware together. Since the display depended on an oscilloscope, it had to be refreshed many times a second through the use of the MP-T interrupt timer.

While programming the driver routine into EPROM, I realized I had no place to put the display data or a few necessary data bytes. There was no problem in RAM, but EPROM can't be changed during execution! I had no room in low memory, which contained the BASIC interpreter, and no room in middle memory, which housed the disk operating system. That left the monitor scratchpad at $A000. But since my RT-68MX monitor uses about three-fourths of it, I was not going to fit over a hundred data bytes into a 128-byte RAM.

The solution was obviously

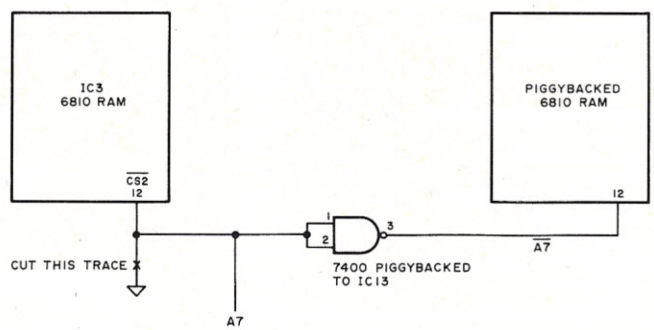


Fig. 2. Wiring the new 6810 RAM to respond to addresses $A080-$A0FF.

more memory. But that involves a capital outlay of at least $100 for a 4K memory board and, since my motherboard is full, a new motherboard and power supply!

Fortunately, for only the small amount of memory I needed, there was an easier solution. I keep spares of all the major chips used in the 6800 system, including a spare 6810 RAM. By integrating this 128 × 8 memory into the system, you can obtain another 128 bytes. The logical place is at address $A080, since the existing 6810 is at $A000-$A07F. The SWTP 6800—on the old MP-A card—uses incomplete address decoding, so the existing 6810 not only responds to $A000-$A07F, but also to $A080-$A0FF, $A100-$A17F . . . all the way up to $BFFF. This 128-byte RAM uses 8K of memory space.

I connected pin 12 of IC3, a 6810, to address line A7. This active-low chip-select pin, tied permanently to ground, responds only when A7 is low, or for addresses $A000-$A07F, $A100-$A17F, $A200-$A27F, etc. Another 6810—otherwise connected identically as IC3, except for pin 12—could be configured to respond to $A080-$A0FF, $A180-$A1FF and so on.

You only need three wires and two ICs to make this change. One IC is a new 6810 RAM, which is piggybacked over the old 6810. All pins, except pin 12, are soldered over the pins of the existing 6810. The 6810s are CMOS chips, so you must observe static precautions. You should also remove the 6800 CPU and 6830 monitor ROM during this operation. Bend pin 12

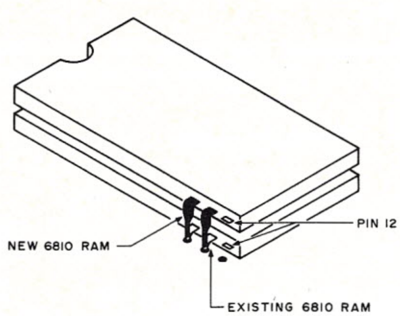


Fig. 1. Method of piggybacking the new 6810 RAM on the MP-A board.

out sideways, and, with a pair of cutters, cut pin 12 of the old 6810 from the PC board and bend it out sideways. Using a short piece of wire, connect address line A7 directly to pin 12 of the old 6810. (See Fig. 1.)

To make the new 6810 respond to the proper addresses, address line A7 must be inverted. Unfortunately, there are no unused gates or inverters on the MP-A card, so you must add one. A low-cost 7400 NAND gate will serve the purpose and be used later.

Piggyback the 7400 over an existing 7400, IC13, but solder down only the power and ground pins, 7 and 14. Bend the other twelve pins of the new 7400 out sideways so they do not touch each other or the board. One gate of the chip is used as an inverter by connecting address line A7 to pins 1 and 2; the gate output, pin 3, is connected to pin 12 of the new 6810. (See Fig. 2.)

Since the 6810 outputs are Tri-states and since all other address and control pins are the same, the two RAMs will now not interfere with one another and will give you a full page of memory from \$A000 to \$A0FF.

Opening Up the Bus

Whenever any memory on the CPU card is accessed, the data bus drivers, IC8 and IC9, are disabled. The MP-A card contains not only the 6810 RAM, but also a 6830 monitor ROM, whether it be MIKBUG, SWTBUG or RT-68MX. To be MIKBUG-compatible, this ROM must respond to addresses from \$E000 to \$E1FF. But to include the interrupt and restart vectors, this same ROM must also respond to addresses \$FFF8 through \$FFFF. This is accomplished by incomplete decoding of addresses, so that the 1K ROM is repeated eight times from \$E000 to \$FFFF. Also, there is the 6810 RAM at \$A000.

The partial address decoding is handled by the upper half of IC16, which disables the bus when both address lines A13 and A15 are ones, or when addresses \$A000–\$BFFF or \$E000–\$FFFF are present. The latter 8K is necessary for the interrupt vectors, but there is no reason for the 128 × 8 (or 256, if the modifications above are made) 6810 RAM(s) to use up 8K of memory.

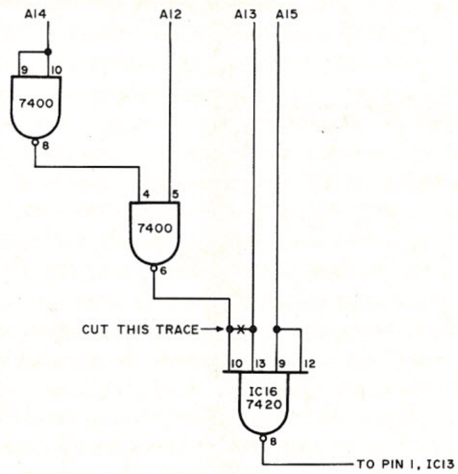


Fig. 3. Modification of MP-A board to open up addresses \$B000–\$BFFF.

With the 7400 in place over IC13, a minor change may be made to allow the data bus to carry information for addresses \$B000–\$BFFF. This is accomplished by letting IC16 disable the bus only when lines A12–A15 are hex A, E or F. That is, when A13 and A15 are high, but when A14 is low and A12 is high, do not disable the bus.

IC16 is a dual four-input NAND gate. The upper half is used as a two-input NAND on A13 and A15. There are two unused inputs. Cut the trace to pin 10 of IC16 and add wires to the piggybacked 7400 as shown in Fig. 3. Now the bus will be disabled when A13 and A15 are high (1s input to IC16). When A12 = 1 and A14 = 0, the input to pin 10 of IC16 will be zero and the NAND will produce a high output, enabling the bus.

This process allows the user to use all of memory space from \$B000 to \$DFFF, adding an extra 4K of accessibility, for a total of 12K.

Open Up the Motherboard

Incomplete decoding plays a part on the SWTP MP-B motherboard as well. The I/O slots respond to addresses \$8000 to \$801F, \$8040 to \$805F, \$8080 to \$807F . . . up to \$9FFF. The blank spaces in between are for future expansion so that a second motherboard may be added.

Again, there is no reason to tie up 8K of memory space for 32 bytes of memory-mapped I/O.

The area from \$9000 to \$9FFF may be opened up for use by a simple expedient. On the motherboard, pin 5 of IC3 is grounded. This is a chip enable pin that must be low to permit the I/O slots to respond. If the trace to pin 5 is cut and a wire is connected from pin 5 of IC3 to address line A12 when address line A12 is zero (\$8000–\$8FFF), the I/O will work. When the line is high, as in \$9000–\$9FFF, the I/O bus will be quiet. This allows the user to place a memory board or EPROM at this location.

For users of the Smoke Signal Broadcasting BFD-68 floppy-disk system, the above-mentioned addition of a second motherboard for more I/O slots is more difficult because the bootstrap ROM on the BFD-68 is configured in a unique way: I/O slots 0 through 7 respond to \$8000 to \$801F, and more, as stated above. The bootstrap ROM "fills in the cracks." That is, it uses the unused locations \$8020 to \$803F, \$8060 to \$807F, and so on. Furthermore, the decoding for these is complete. The 1K ROM simply uses every other 32 bytes to load the disk operating system so that no additional memory is used.

There is no room for adding a second motherboard if more I/O is needed. Since the I/O takes the first 32 bytes, the disk the second 32, the I/O the next 32 and the disk the fourth 32, there is no room left all the way to \$8FFF (or \$9FFF, if the above change is not made).

If the second motherboard could be made to respond to \$8040 to \$805F, and the first motherboard not to respond to

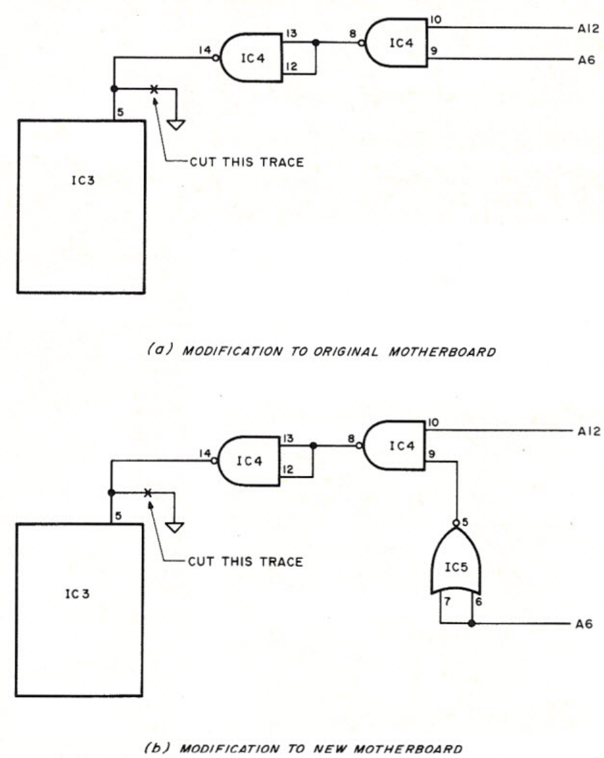


Fig. 4. Rewiring motherboard so that the I/O slots will respond to a) \$8000–\$801F, b) \$8040–\$805F. Also, opening up addresses \$9000–\$9FFF.

these addresses, this would solve the problem. Fortunately, you can do this without additional ICs.

Place address line A5 low for I/O and high for the DOS bootstrap. If you desire to decode address line A6 and if A6 = 0, I/O will respond to addresses $8000 to $801F. However, if A6 = 1, I/O will respond to addresses $8040 to $805F. IC4 and IC5 are quad two-input NAND and NOR gates, respectively. Between them, there are three unused gates, which may be used for additional address decoding. We can still use the A12 line to open up the $9000-$9FFF addresses.

The unused gates are disabled by grounding the inputs. With a sharp razor blade, cut the foils connected to IC4, pins 9 10, 12 and 13, on the bottom of the board so that each pin is unconnected. The same is true for pins 5 and 6 of IC5. Then run jumper wires from address lines A6 and A12 to these ICs and to pin S of IC3, as shown in Fig. 4. Be sure to cut the ground line foil from pin 5 of IC3, as above, on the bottom of the board. Fig. 4a shows the modification for the first motherboard to let I/O respond to $8000-$801F; Fig. 4b shows the modification for the second motherboard to allow I/O to go out at $8040-$805F.

Finally, let me caution you about opening up $9000-$9FFF. Users of the BFD-68 disk system should be aware that the disk drive is interfaced through a PIA that is addressed at $9FFC. Any attempt to put a full 4K of memory into $9000-$9FFF will interfere with this PIA, causing problems with the disk.

Summary

For the price of two integrated circuits and some wire, you can upgrade your SWTP 6800 in about an hour's time. This results in the addition of another 128 bytes of memory, opening up 8K of memory space and the addition of a second motherboard without interfering with the BFD-68 disk system. That's what I call cost-effective work! ■

The New CP/M: Is It Worth It?

This reviewer thinks so—and lists some reasons why.

Richard Fritzson
25 Callodine Ave.
Amherst, NY 14226

Digital Research has released an updated version of their popular CP/M microcomputer operating system. The old one was release 1.4; the new one is release 2.0. First-time purchasers of CP/M have no choice; only the most current release is sold. However, those who already have CP/M now have to decide whether to spend the money needed to update their system.

New Features

CP/M 2.0 is user configurable to all types of currently available disk drives: floppies and mini-floppies, including single, double and quad density, as well as the new hard disks. If you plan to upgrade your hardware to include a different type of disk than you have now, you need the new CP/M.

Externally, CP/M 2.0 has several new conveniences. The console I/O routines now recognize the back space and delete characters on CRT terminals instead of echoing them (the rubout key still deletes and echoes). Each individual file on a disk can be marked as "read-only," preventing its modification, renaming or accidental erasure.

Files can also be marked as "system-files," which means only that they will not appear in a directory listing. This apparently reduces the clutter of having the standard .COM files appearing in each directory listing, which is now four columns across instead of only one.

Each CP/M disk can be shared by up to 16 users, each of whom is assigned a number; all file access is restricted to one user number at a time. This feature is convenient if you share your large, hard-disk system with others, but I don't think floppy disk users will have much use for this feature. Since every user needs his own copy of STAT, PIP, ED and all the other standard utilities, disk space disappears rapidly.

The most noticeable improvement in this version is on the inside. CP/M finally has decent random access capability for its disk files. This means that you can read or write each of up to 65,536 128-byte records, in any order, by simply giving a record number and making an I/O request. There is no need to calculate which extent the record is in, then open and position the extent before reading. There is no need to write to and allocate all of the records prior to the one you want to write.

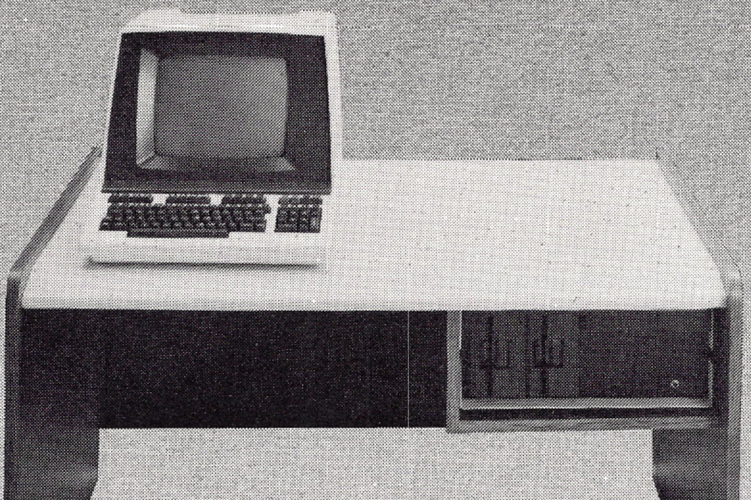
CP/M only allocates disk space when you actually write a new record. You can write the first and last records of an eight-million-byte file on your mini-floppy; but don't try to fill in all of the space in between.

Old vs New

If you're an assembly-language programmer who does a lot of disk I/O, then the simplified access to files and the "virtual file" facilities are a real help. But if you run mostly prepackaged software or if you write BASIC programs under an interpreter that runs under your current CP/M, then all you gain are the amenities, such as four-column directory listing and individual read-only file.

I suggest that you wait a while before purchasing an updated CP/M. Let Digital Research eliminate the bugs (there were some in the first one I tried; they're gone now), or wait until they release version 2.1.

However, don't be surprised when the next piece of software you want —whether it's a language processor such as the latest BASIC or Pascal or a system tool such as a text editor or, especially, a data base package—requires the new CP/M, or at least tempts you by promising improved performance if you have it. The assembly-language programmers who write those will be using the new file access features. ■



Dial-up Directory

Communications software and the OSI C1P are the subjects of this Directory.

Frank J. Derfler Jr.
PO Box 691
Herndon, VA 22070

This installment of Dial-up Directory will deal with the hardest part of computing: the software. I will review some old and new communications software and devote a special section to the telecomputing uses of the OSI C1P.

Old Software Revisited

I reviewed Leonard Garcia's Telestar program a few months ago (March 1980, p. 50). This North Star communications software has been improved and expanded. Telestar is designed specifically for the Horizon computer, but it will run on any North Star system with the proper addressing. The original program turns your computer into a smart terminal capable of saving or recording a complete telecommunications session on disk and calling it out for review, printing or retransmission.

The new version provides even greater flexibility. Incoming data can now be designated to be saved or not saved (this conserves space in both RAM and disk). Another new feature allows you to send a file to a timesharing or other system and to receive data to memory and disk. Previously, this could only be done with another Telestar user.

Leonard includes another program called Telshare, which allows simultaneous use of a Horizon by a local and remote input. You can operate the system locally while another person either monitors or works along with you from a remote location. This is particularly well suited for demonstrating programs or for taking turns playing games. It also has potential for tutoring during computer-assisted instruction. Leonard includes six versions of the program to suit different RAM sizes.

This program, like Telestar, has a built-in customizing routine. Both programs are meant to be used with the standard RS-232 ports and an external acoustic modem, but Leonard supplies software listings to interface with the Potomac Micro-Magic modem board. Both programs are well documented and cost $30. Contact Leonard E. Garcia, 3517 Herschel Rd., Dallas, TX 75219.

Another revised software package came in from The Microstuf Company in Decatur GA. Their Remote Console and Bulletin Board programs are used on the Remote North Star message system in Atlanta, (404) 939-1520.

Remote Console is available for all North Star and most CP/M systems. This machine-language program interfaces a D. C. Hayes or Potomac Micro-Magic auto-answer modem to the disk I/O routines for true auto-answer capability. It provides the automatic baud rate selection, call counter, boot-up and other functions. The standard version loads at 0100H.

Bulletin Board is a package of BASIC programs that provides a complete electronic bulletin board service including a help file, menu, logging, message field search, message file, program file and system operator editing functions. This means that if you have some other phone line answering arrangement, the bulletin board software is still available in BASIC for your use. The package costs $30, not including various options and customizing costs.

Microstuf's XLINK and CLINK are smart terminal and file transfer programs for North Star and CP/M, respectively. They are meant for use with the D. C. Hayes modem and allow transfer of files, auto-dialing and various file output and reading capabilities. These programs tie the disk, modem and computer together into one effective smart terminal package. They cost between $25 and $50, depending on the versions.

Documentation and utilization are important parts of any software package. The documentation for Len Garcia's Telestar is excellent. Len supplies program listings, comments and patch points, plus many practical hints on operation. The programs also have a self-patching capability in which they cue the user through changes and then save themselves on

disk in the changed version.

Microstuf's documentation is less complete, but their philosophy is different. Les Freed and Bob Strong, the Microstuf folks, work with their customers to provide a package as "pre-customized" as possible. The programs written in BASIC are self-prompting and clear. The documentation is sparse, but the service is great. If you order from Microstuf, you should include the detailed specifications of your system such as addresses, memory size and keyboard/terminal arrangement.

Talk on Your Apple

Apple computers make good terminals for data communications. Several software packages are available to meet different hardware configurations. The ultimate sparkle for a highly polished Apple comes with Bill Blue's Apple software package called ASCII Express. This package of six programs needs 48K, Applesoft in ROM, at least one disk drive and a D. C. Hayes MicroModem to run. It allows you to transfer programs and files to and from *any* other system.

Along with the file transfer program, Bill has included routines to make Applesoft and integer programs into textfiles, compress files for more economical transmission over long-distance phone circuits, read textfiles locally, create textfiles from blocks of binary data and operate the Apple as a terminal in uppercase and lowercase (with the Don Paymay lowercase board). The disk Bill sends is packed full. Documentation for the ASCII Express is descriptive.

This software is set for a fairly specific machine configuration, but when I demonstrated it to several local Apple users, they pointed out sections that would also be valuable to people using a communications card and external modem. Bill Blue and Craig Vaughan are the authors of the ABBS software. Bill runs the People's Message System at (714) 449-5689. The complete ASCII Express (Bill Blue, PO Box 1318, Lakeside, CA 92040) sells for $34.95.

The people who run The Source information utility have an Apple II program specifically designed for their system format. After you bootup the program, it will dial The Source automatically (if you have a D. C. Hayes board), sign you on the system and transfer files and data between your system and The Source's Prime computers. It will work without the D. C. Hayes board, but you have to do the dialing.

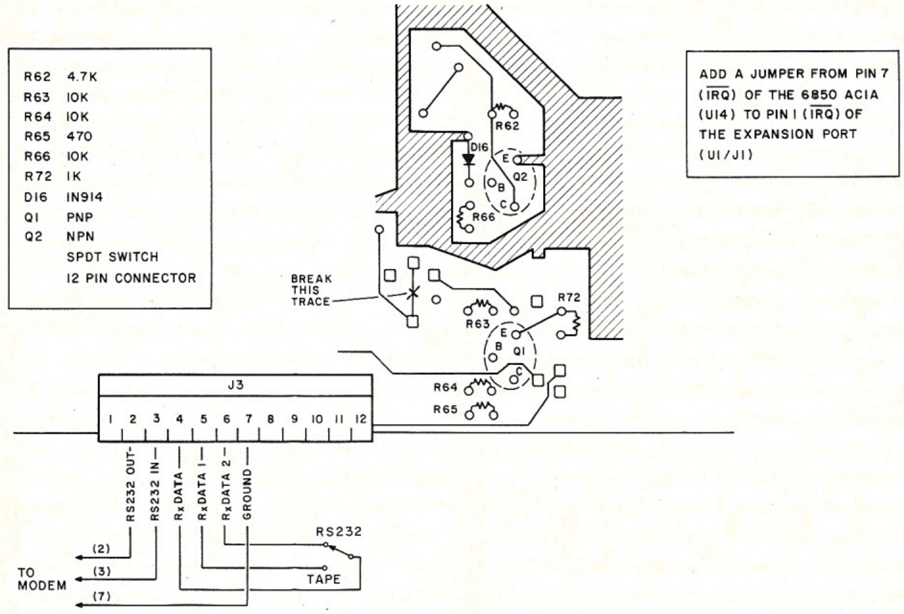


Fig. 1. Parts you must add to the OSI C1P circuit board to get RS-232 output. The transistors are not critical. A 2N2222 will work fine for the npn transistor, and a 2N2905 or similar transistor will serve for the pnp transistor. Radio Shack 276-2033 and 276-2034 will work too.

Because the Apple displays fewer characters per line than most systems, a special feature that scans the incoming data and prevents the "breaking" of words at the end of a line by automatically moving the word to the next line is included. This Source Apple II Terminal Package is $29.95 from Telecomputing Corporation of America, 1616 Anderson Rd., McLean, VA 22102.

If you don't want to buy a D. C. Hayes modem board, you can use the Apple computer as a terminal for just a few dollars. Many small enterprises are making communications boards for the Apple. A communications card used with an outboard acoustic coupler works OK. Electronic Systems makes an RS-232 card that is available as either a kit ($42) or assembled ($62). A program listing that comes with the card allows elementary operation as a terminal. Some patching and personalizing may be needed, so this is not a project for novices, but the price is certainly right.

Bill Hyde of San Antonio has come up with a program that is not purely for on-line telecommunications, but it sure is a convenient aid. Bill's Autodialer provides the Apple user with a listing of bulletin board, time-sharing, digital network and other telecomputing systems. You can easily change and add to all data, including remarks. You can review the listings, numbers and comments, and (if you have a D. C. Hayes MicroModem) the system will automatically dial the number of the listing you select and turn you over to whatever communications program you may have.

The various transmission options of the D. C. Hayes board are software selected through the program, which will work on disk or tape; either version costs $15. Bill runs the San Antonio ABBS, (512) 657-0779, or he can be reached at PO Box 12883, San Antonio, TX 78212.

OSI C1P

The OSI C1P provides economical, but versatile, computer power. *Microcomputing* (January 1980, p. 140) ran a review of the C1P in which the author speculated that this system could be used as a low-cost terminal for telecomputing use. He was right, but first somebody had to write the software and define the hardware changes needed. Enter Mike Carroll of Tulsa OK. Mike has provided us with the hardware changes and software listing to use the C1P as a dumb terminal. Smart terminal routines are in the works.

Any computer acting as a dumb terminal has only two jobs:

1. Any data entered by the local keyboard must be sent to a UART (universal asynchronous receiver/transmitter), where it is converted to a string of serial bits and sent to a modem in the RS-232 format. RS-232 is a ± 12 volt signaling scheme that is

the most common standard for serial transmission in the microcomputer world. In the OSI C1P the UART is a Motorola 6850. Motorola calls their chip an ACIA, which is how it is referred to in the program comments.

2. Any data received from the modem via the UART must be displayed on the screen.

Dumb terminal is an appropriate name for such a computer. After the data leaves the screen, it is gone for good. Smart terminals can save the data and recall it for later use.

On other computers, the terminal program alternately reads the keyboard and UART ports and routes the data to the right place. The OSI C1P does not work this way. It uses a polled keyboard scheme in which software loops monitor when a key is pressed and determine which one it is. Unfortunately, this means that data from the UART may go unnoticed and become lost while the monitor is working with the keyboard. This program uses the interrupt capability of the 6502 CPU to get the attention of the monitor.

The terminal program for the C1P calls the monitor routine to scan the keyboard. When a keyboard character is received, it is written to the UART through the program and the monitor is called again. This is an endless loop. When the UART receives a character in from the modem, it interrupts the keyboard polling and calls for the program routine to send the character to the screen. The keyboard scan then resumes.

Some parts and hardware changes are needed to operate the C1P as a terminal. A hardware jumper must be added so that the interrupt can reach the 6502 (see Fig. 1). This addition consists of a jumper from pin 7 of the 6850 to pin 1 of the expansion port. The C1P does not come with full RS-232 capability. The circuit board traces are present, but some parts are needed. These parts are described in some of the OSI documentation, but not clearly. The parts list and circuit diagram of Fig. 1 show where they should go. It is not a difficult modification, but one that was left to the user to keep the initial cost down.

```
ADDR  OBJECT    LABEL   COMMAND          COMMENTS
----  --------  ------  ---------------  ------------------------------
                *
                *  'Dumb' terminal program.
                *     M. Carroll  10 January 1980
                *
                *  Set IRQ vector so that our routine gets control.
                *
0300  A2 02     START   LDX #$02         Copy jump command
0302  BD 27 03  VSET    LDA JMP,X          to the IRQ
0305  9D C0 01          STA $01C0,X          vector
0308  CA                DEX
0309  10 F7             BPL VSET
                *
                *  Reset ACIA and configure it for 7 data bits,
                *  even parity, 1 stop bit, and allow 6850 to interrupt
                *  when a character is received.
                *
030B  A9 03             LDA #$03         reset ACIA
030D  8D 00 F0          STA $F000
0310  A9 8D             LDA #$8D         configure ACIA
0312  8D 00 F0          STA $F000
                *
                *  Only now do we allow IRQ's to take place
                *  (could otherwise cause loops from spurious
                *  interrupts.)
                *
0315  58                CLI              allow IRQ'S
                *
                *  Get a key from the monitor keyboard scan routine.
                *  When a key is pressed, wait for ACIA XMIT buffer
                *  to empty and then send key to modem.
                *
0316  20 ED FE  READ    JSR $FEED        get a key
0319  48                PHA              save it
031A  AD 00 F0  RDLOOP  LDA $F000        get ACIA status
031D  29 02             AND #$02         check XMIT ready
031F  F0 F9             BEQ RDLOOP       not ready, loop back
0321  68                PLA              reload key
0322  8D 01 F0          STA $F001        give to ACIA
0325  D0 EF             BNE READ         get another key
                *
                *  This command is copied to the IRQ vector.
                *
0327  4C 2A 03  JMP     JMP WRITE        IRQ action
                *
                *  This routine gets control via the IRQ vector
                *  whenever the 6850 receives a character and
                *  interrupts the CPU.  Since the 6850 will interrupt
                *  when the character is ready, we don't have to
                *  check the ACIA status.  Pull the character from the
                *  ACIA and give it to the monitor routines to be
                *  written on the screen.  Return to the interrupted
                *  routine (most likely the keyboard scan.)
                *
032A  48        WRITE   PHA              save reg
032B  AD 01 F0          LDA $F001        get char from ACIA
032E  20 2D BF          JSR $BF2D        write it to screen
0331  68                PLA              restore reg
0332  40                RTI              return
```

Program listing. Polls the keyboard, sends interrupts to the CPU when the UART receives data from the modem and displays the received characters on the screen. Note that all received characters are displayed including control codes and anything else sent by remote systems.

It is important to note that even when the parts are added and the circuit trace is cut, the output voltage does not meet the full RS-232 standard. No negative voltages are used in the C1P's 6502 system, and the highest voltage available is +5 volts. Therefore, the output swing is only from 0 to +5 volts instead of the higher plus and minus swings of true RS-232.

Many modems will work with these voltages, but if your modem will not, then you can put a separate board such as the Electronic Systems TTL/RS-232 board (about $10) between the computer and the modem. This board could use power borrowed from the modem. The C1P RS-232 modification will accept the full RS-232 voltage swing on the input.

The C1P runs at about 320 baud instead of 300 baud because of various money-saving frequency sources and dividers used. This has not been a problem in operating with Tymnet, most bulletin boards and IBM-equivalent communications equipment. The displayed line length is 24 characters, and 24 lines will fit on the screen before they move off. This is not a major drawback if the control-S or pause command is used to hold the input from a system while a note is made or the text reread.

The OSI C1P retails for about $350. You must add a tape recorder and monitor, but the entire system could be on the air for under $500, which is several hundred dollars under any other sort of terminal capability. Who is going to bring up the first OSI Bulletin Board System? OSI C1P users who have comments or questions can write to Mike Carroll at PO Box 2844, Tulsa, OK 74101.

Dial-up

Many different computers can be used as terminals. There are definite cost, capability and convenience trade-offs between the systems, but no matter what system you have, using your computer to communicate can open new worlds of education, information and interest.

I would like to hear your comments, questions and experiences. Tell me about hardware and software you have used for telecommunications. Send paper mail to PO Box 691, Herndon, VA 22070. Include an SASE if you want a reply. Send electronic mail to the Remote North Star (404) 939-1520 or to TCB967 on The Source. ■

TRS Text Formatter

Modify Law and Mitchell's formatter (May '79) for TRS-80 use.

L. H. Daniels
12674 Forestedge
St. Louis, MO 63138

As those of you with TRS-80s who have attempted to use Law and Mitchell's excellent "Editor/Formatter" program (*Microcomputing*, May 1979, pp. 26–33) already know, the program requires some alterations before it will run in your Level II machines (see Table 1).

With these changes accomplished, you sat down to do some word processing you had been putting off until you had such a program and immediately encountered . . . .

Disappointment #1

The program worked beautifully as long as you didn't insert any commas, quotes or colons in your text material. Commas, quotes and colons are, of course, string delimiters in Level II BASIC. After receiving a few EXTRA IGNORED error messages, you learned that you could at least use commas and colons if you remembered to lead off each line insertion with quote marks, and resigned yourself to using an apostrophe as a quote-marks substitute. That's the cue for . . . .

Disappointment #2

You type in a page or two of text and want to add more but don't have the time, so you decide to save it on tape. Later, you return, load the text back into memory and begin to review it. You are surprised to find that whole sentences are missing or have been reduced to a few words. No problem, you say; I'll just load the other copy (you always make at least two copies of everything). But this yields the same results. What happened?

Those commas and/or colons sprinkled throughout your text were treated as string delimiters by the PRINT#−1 statement when the data was written to tape—in spite of the leading quote marks (and with no error messages to warn you of what was happening). Now, you must choose between using civilized punctuation or being able to save text data on cassette.

Not so. The following will describe a machine-language program which is called by the "Editor/Formatter" program that will:

1. Permit you to use any printable ASCII character in your data set.
2. Input/output your text data on cassette tape exactly as it was originally written.
3. Familiarize you in a very practical way with the seldomly used VARPTR and USR (0) functions.

Line Input Using INKEY$

First, before I describe this program, I'll show you a way to eliminate disappointment #1 with just a few more changes and some added program lines. This is a quick modification that will fit the need of those who want to use commas, quote marks and colons but don't require their data to be saved on tape.

The fortunate users of Radio Shack Disk BASIC will recognize immediately that what we Level IIers need is their LINE INPUT function. The Disk BASIC LINE INPUT statement allows the user to input a string variable, which is terminated only

```
25 L=L(M):P=1:I$="": PRINT "COMMAND ?";:GOSUB5000:PRINT:
   IFI$=""THENI$=B$:GOSUB1000:ELSEGOSUB1000:IFR<0THEN 40

124 I$="":PRINT">?";:GOSUB5000:PRINT:IFI$=""THEN2100


4999 REM - BUILD I$ WITH INKEY$
5000 PRINT CHR$(14);
5010 A1$ = INKEY$ : IF A1$ = "" THEN 5010
5020 IF A1$ = CHR$(13) THEN RETURN
5030 IF A1$ = CHR$(8) AND LEN(I$) = 0 THEN PRINT " ";
     :PRINT A1$; : GOTO 5010 ELSE IF A1$ = CHR$(8) THEN
     I$ = LEFT$(I$,LEN(I$)-1) : PRINTA1$; : GOTO  5010
5040 IF A1$ = CHR$(24) AND LEN(I$) = 0 THEN 5010 ELSE
     IF A1$ = CHR$(24) THEN FOR I = 1 TO LEN (I$)
     : PRINT CHR$(8); : NEXT : I$ = "" : GOTO 5010
5050 IF A1$ = CHR$(9) THEN  A1$ =
     STRING$(8 - 8*(LEN (I$)/8 - INT(LEN (I$)/8)),"  ")
5060 IF LEN(I$) + LEN(A1$) > 255 THEN RETURN
     ELSE I$ = I$ + A1$ : PRINT A1$; : GOTO 5010
```

Modification A. Changes in "Editor/Formatter" lines 25 and 124 and the addition of a new subroutine at line 5000 to provide the LINE INPUT function by using BASIC's INKEY$ statement.

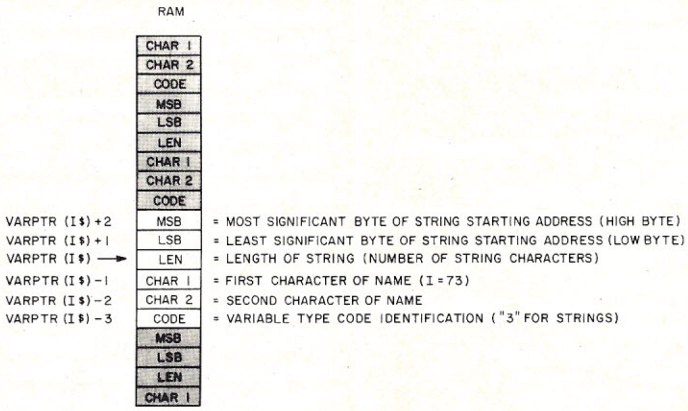


Fig. 1. VARPTR (I$) = the memory address that contains one byte of information about I$. Note that by incrementing or decrementing the VARPTR values, the other bytes can be accessed. Shaded areas represent the continuation of the variable table.

by a carriage return (pressing the ENTER key generates a carriage-return character).

By retyping lines 25 and 124 of the original program and adding an INKEY$ subroutine at line 5000 (see Modification A), you can duplicate the LINE INPUT functions.

Because the familiar question mark (?) prompt generated by the BASIC INPUT statement is lost with the use of the INKEY$ subroutine, "Command ?" and ">?" are printed to take its place. I inserted the word "Command" to help me distinguish between the COMMAND and INSERT modes of the program.

If you use Modification A, when a command or text string input is required, a prompt will be displayed and the subroutine beginning at line 5000 will be entered.

Line 5000 turns on the cursor. The program then loops in line 5010 until you press a key. When you do, A1$ is set equal to the value of this character. Line 5020 checks to see if the key you pressed was ENTER, and if so, it will return with I$.

Lines 5030 and 5040 reposition the cursor and prevent wiping out your prompt symbol when the control functions of back space (Arrow Left) and clear-to-the-beginning-of-line (Shift, Arrow Left) are used. Line 5050 enables the TAB function (Arrow Right).

Finally, in line 5060, after a check is made to keep I$ from exceeding 255 characters, A1$ is concatenated (love that word!), or added, to I$, displayed on the monitor, and the GOTO 5010 causes the whole process to repeat. Thus, I$ grows character by character until *you* terminate it with the ENTER key.

Machine-Language Mod

As mentioned, the INKEY$ subroutine, though useful for some, doesn't put us any closer to being able to save/load delimited string data. However, Program 1 (PGM 1) is the source listing of a machine-language program that can be placed in high "protected" memory to enable the "Editor/Formatter" to work as Messrs. Law and Mitchell intended.

If you're not familiar with assembly-language source code or machine-language instructions, don't be discouraged. Just make the changes to the original program lines and type in the new program and DATA lines in the Modification B listing.

Thereafter, when you CLOAD the "Editor/Formatter" program and enter RUN, the machine-language code contained in the DATA lines will be POKEd into memory and the BASIC program/DATA lines involved in this POKEing will be automatically deleted, so they don't take up memory space.

Modification B

Line 1. Except for the CLEAR 4500 and UN = 1 statements, original line 1's function was to set certain of the program variables to zero. Because Level II BASIC does this for us each time we enter RUN, only these two statements were saved and moved to line 2 to free line 1 as a "jumping off" place.

The GOSUB 4000 causes a branch to the subroutine that set up the USR(0) entry address, installs PGM 1 into memory and prints a message to video reminding you to delete line 1. When control returns to line 1, lines 4000 through 4080 are deleted to regain memory space. The execution of this last instruction puts you back into the BASIC COMMAND mode with the message printed by the subroutine still displayed.

A better way to accomplish the above would be to put the loading subroutine at the very beginning of the main program. With all lines contiguous, it could delete itself completely. But what a job it would be to clear enough room to do this with the "Editor/Formatter"!

Line 25. The original INPUT I$ of line 25 is replaced by the statements beginning with PRINT "COMMAND ?" and ending with AA% = USR(0). PRINT "COMMAND ?" is our prompt symbol.

AA% = VARPTR(I$) sets AA% equal to the decimal address where BASIC keeps its information about I$—its variable-type identification code, name, length and the memory address where the I$ characters are actually stored. If you think of VARPTR(I$) as pointing to the I$ place in BASIC's "Table of

1. Change all PRINT commands in lines 1200 through 1216 to LPRINT to enable line printing.
2. Replace the statement PRINT SPC(K) in line 1206 with LPRINT STRING$ (K,CHR$(32)) to gain the benefit of the Formatter's random-insertion-of-spaces routine.
3. Rewrite lines 700 through 781 for cassette tape input/output.
4. Change the Formatter control character in line 4 from G$ = "\" to G$ = "↑" (or other character of your choice), because the TRS-80 character set doesn't include a backslash.
5. Change all references to RND(1) to RND(0).

Table 1. Minimal changes required to run the "Editor/Formatter" program in TRS-80 Level II BASIC.

Program 1. Assembly-language source listing in Zilog mnemonics.

```
              ;******************************************************
              ;* "LINEIN" - ROUTINE TO DUPLICATE THE LINE INPUT *
              ;* FUNCTION OF DISK BASIC AND PROVIDE TAPE SAVE/  *
              ;* LOAD FOR STRING DATA CONTAINING DELIMITERS.    *
              ;*                                                *
              ;*        L. H. DANIELS                           *
              ;*        MAY 3, 1979                             *
              ;******************************************************
              ;
7DB5                        ORG     7DB5H
7DB5 2A8E7E   BEGIN         LD      HL,(VARPTR)  ;GET POINTER ADDR
7DB8 11907E                 LD      DE,STRING    ;DE POINT TO STRNG STORE
7DBB AF                     XOR     A            ;CLEAR
7DBC 77                     LD      (HL),A       ;MAKE STRING LEN 0
7DBD 23                     INC     HL           ;  AND PUT ADDR OF
7DBE 73                     LD      (HL),E       ;  STRING STORE INTO
7DBF 23                     INC     HL           ;  BASIC PGM'S POINTER
7DC0 72                     LD      (HL),D       ;  TO STRING
7DC1 2B                     DEC     HL           ;ADJ TO POINT TO
```

```
7DC2 2B      00290        DEC    HL           ;  ADDR OF STRING LEN
7DC3 3E0E    00300        LD     A,0EH        ;CURSOR BYTE
7DC5 CD0D7E  00310        CALL   DISPLY       ;TURN ON CURSOR
7DC8 D5      00320 REPEAT PUSH   DE           ;SAVE
7DC9 CD2B00  00330 AGAIN  CALL   2BH          ;KEYBRD SCAN
7DCC B7      00340        OR     A
7DCD 28FA    00350        JR     Z,AGAIN      ;REPEAT IF NO CHARACTER
7DCF D1      00360        POP    DE           ;RESTORE
7DD0 FE0D    00370        CP     0DH          ;CARRIAGE RETURN ?
7DD2 2839    00380        JR     Z,DISPLY     ;  YES, DISPLAY & RETURN
7DD4 FE01    00390        CP     1            ;BREAK KEY ?
7DD6 C8      00400        RET    Z            ;  YES, RETURN
7DD7 FE08    00410        CP     8            ;BACKSPACE ?
7DD9 2820    00420        JR     Z,ONEBK      ;  YES, BACKUP CURSOR
7DDB FE18    00430        CP     18H          ;CANCEL ?
7DDD 2819    00440        JR     Z,POLYBK     ;  YES, CLR LINE
7DDF FE09    00450        CP     9            ;TAB?
7DE1 2830    00460        JR     Z,TAB        ;  YES, GO TAB
7DE3 FE20    00470        CP     20H          ;IGNORE ALL OTHER
7DE5 38E1    00480        JR     C,REPEAT     ;  CONTROL CHARACTERS
7DE7 CDEE7D  00490        CALL   BUILD        ;ADD CHAR TO STRING
7DEA 20DC    00500        JR     NZ,REPEAT    ;IF LEN <255 GO FOR MORE
7DEC 180D    00510        JR     ONEBK        ;IF > 255 BACK UP ONE
7DEE 12      00520 BUILD  LD     (DE),A       ;STORE CHARACTER
7DEF CD0D7E  00530        CALL   DISPLY       ;  AND DISPLAY
7DF2 13      00540        INC    DE           ;BUMP POINTER
7DF3 34      00550        INC    (HL)         ;ADD 1 TO STRING LEN
7DF4 7E      00560        LD     A,(HL)       ;COMPARE STRING LEN
7DF5 FEFF    00570        CP     0FFH         ;  TO MAX OF 255
7DF7 C9      00580        RET
7DF8 46      00590 POLYBK LD     B,(HL)       ;# OF BACKSPACES INTO B
7DF9 1802    00600        JR     ZEROCK
7DFB 0601    00610 ONEBK  LD     B,1          ;1 FOR 1 BACKSPACE
7DFD 7E      00620 ZEROCK LD     A,(HL)       ;CHECK FOR STRING LEN
7DFE FE00    00630        CP     0            ;  EQUAL TO ZERO
7E00 28C6    00640        JR     Z,REPEAT     ;  GO IF YES
7E02 3E08    00650 BKSPC  LD     A,8          ;BACKSPACE CNTRL INTO A
7E04 CD0D7E  00660        CALL   DISPLY       ;DISPLAY
7E07 1B      00670        DEC    DE           ;ADJUST STRING STORAGE
7E08 35      00680        DEC    (HL)         ;  AND STRG LEN FOR BKSP
7E09 10F7    00690        DJNZ   BKSPC        ;REPEAT FOR VALUE IN B
7E0B 18BB    00700        JR     REPEAT       ;GO FOR ANOTHER CHAR
7E0D D5      00710 DISPLY PUSH   DE           ;SAVE
7E0E CD3300  00720        CALL   33H          ;BASIC ROM DISPLY ROUTNE
7E11 D1      00730        POP    DE           ;RESTORE
7E12 C9      00740        RET
7E13 7E      00750 TAB    LD     A,(HL)       ;STRING LEN INTO A REG
7E14 D608    00760 TAB1   SUB    8
7E16 3802    00770        JR     C,GO         ;JUMP IF < 8
7E18 18FA    00780        JR     TAB1         ;JUMP IF > 8
7E1A ED44    00790 GO     NEG                 ;MAKE DIFF POSITIVE
7E1C 3C      00800        INC    A            ;ADD 1
7E1D 47      00810        LD     B,A          ;SPACES TO TAB INTO B
7E1E 3E20    00820 TAB2   LD     A,20H        ;SPACE CHAR INTO A
7E20 CDEE7D  00830        CALL   BUILD        ;ADD TO STRING STORAGE
7E23 28D6    00840        JR     Z,ONEBK      ;JUMP IF STRING LEN=255
7E25 10F7    00850        DJNZ   TAB2         ;REPEAT FOR VALUE IN B
7E27 189F    00860        JR     REPEAT       ;GO FOR ANOTHER CHAR
             00870 ;
             00880 ;BEGIN TAPE WRITE ROUTINE
             00890 ;
7E29 3A907F  00900 WRITE  LD     A,(FLAG)     ;FLAG VALUE INTO A
7E2C FE01    00910        CP     1            ;IF = 1
7E2E 2810    00920        JR     Z,WRITE1     ;  WRITE TAPE
7E30 FE02    00930        CP     2            ;IF = 2
7E32 282E    00940        JR     Z,UBER       ;  TURN OFF CASSTT & RET
7E34 3E01    00950        LD     A,1          ;SET FLAG
7E36 32907F  00960        LD     (FLAG),A     ;  EQUAL TO 1
7E39 AF      00970        XOR    A            ;CLEAR
7E3A CD1202  00980        CALL   212H         ;TURN ON CASSETE
7E3D CD8702  00990        CALL   287H         ;WRITE LEADER
7E40 2A8E7E  01000 WRITE1 LD     HL,(VARPTR)  ;ADDR STRNG PNTR IN HL
7E43 46      01010        LD     B,(HL)       ;STRING LEN INTO B
7E44 23      01020        INC    HL           ;BUMP
7E45 5E      01030        LD     E,(HL)       ;STRING ADDR INTO DE
7E46 23      01040        INC    HL
7E47 56      01050        LD     D,(HL)
7E48 1A      01060 WRITE2 LD     A,(DE)       ;CHAR TO OUTPUT INTO A
7E49 13      01070        INC    DE           ;BUMP
7E4A CD6402  01080        CALL   264H         ;OUTPUT TO TAPE
7E4D 10F9    01090        DJNZ   WRITE2       ;REPEAT FOR LEN OF STRNG
7E4F 3E0D    01100        LD     A,0DH        ;CARRIAGE RET INTO A
7E51 CD6402  01110        CALL   264H         ;OUTPUT TO TAPE
7E54 0610    01120        LD     B,10H        ;OUTPUT 16 ZEROS
7E56 AF      01130 WRITE3 XOR    A            ;  TO TAPE
```

Variable Listings," it will help to make this more understandable (see Fig. 1).

PGM 1 needs the address of I$'s "Table Listing." So, VARPTR (I$) is converted to two decimal numbers, one equal to the high byte and one equal to the low byte of the corresponding value of VARPTR(I$)'s address in hexadecimal (hex) and POKEd into two consecutive spots in memory where PGM 1 "knows where to look for them" by:

POKE 32399,INT(AA%/256):POKE32398, 256*(AA%/INT(AA%/256)).

The statement, AA% = USR(0), is the signal to BASIC to take a rest and turn control over to PGM 1. To use the USR(0) function, you must first tell BASIC where to jump by POKEing the entry address of your machine-language program into RAM locations 16526 and 16527, again, as two decimal numbers that respectively equal the low and high bytes of the program entry address in hex. This is done for you during "Editor/Formatter" initialization by line 4000.

If the above seems a little complicated, it's only because it is. However, as with many other things, the more you use it the easier it becomes to understand and apply.

PGM 1

Now that the USR (0) has taken us to PGM 1, let's talk for a moment about what goes on up there at the speed of light.

Basically, the "front end" of PGM 1 does the same job as the INKEY$ subroutine, i.e., manages the cursor position and puts your string together character by character. But, it's *where* this string is stored that's different.

Remember that PGM 1 is on its own. BASIC isn't moving things around to make room for a string that has grown too long to fit the initial space allotted it, or any of the hundred-and-one other "housekeeping" chores we can take for granted with BASIC on the job. Instead, that-which-is-to-become I$ is built and saved in a special storage area labeled, appropriately enough, "STRING."

When you push ENTER, which signals the end of your string input and calls for a return to BASIC, PGM 1 first places the beginning address of its string storage space into BASIC's "Table Listing" for I$. In this way, we cause BASIC to "forget" the old location of I$ and to look for it instead in PGM 1's string storage area.

What about I$ length? As your string characters were added (or deleted), PGM 1 kept a tally at the VARPTR (I$) location. Hence, this information is already in place even before insertion of the I$ "pseudo address," which completes the "Table Listing" and provides BASIC with everything it needs to know about I$.

More on Mod B

Line 124. Here, your text strings are input into the "Editor/Formatter." Interaction with PGM 1 to accomplish this is the same as in line 25.

Lines 700 through 791. BASIC and PGM 1 routines are both used to save and load program variables. BASIC's PRINT #-1 and INPUT #-1 are for numeric values and PGM 1's WRITE and READ are for string values.

Because your text data is stored in the L$(n) array, to save it, VARPTR (L$(n)) is passed along to PGM 1 (just as it was for I$). Each element of the array is located in this fashion and subsequently transcribed.

During a load operation, I$ is repeatedly formed in PGM 1's string storage space, using the tape data as a source. Then each element of the L$(n) array is, in turn, set equal to the I$.

Note that lines 711 and 771 can be shortened to:

```
711 FORI=0TO59:INPUT#-1,L(I),F(I):NEXT
771 FORI=0TO59:PRINT#-1,L(I),F(I):NEXT
```

This will gain some extra memory space. I chose the longer form to help speed up cassette I/O operations.

User Instructions

The only new instruction you have to remember is to answer the MEMORY SIZE ? question with 32180. This protects PGM 1 from destruction by BASIC.

Also, don't be surprised when you press the BREAK key and

```
7E57 CD6402                CALL    264H         ;  AS DATA SPACERS
7E5A 10FA                  DJNZ    WRITE3
7E5C 3EA5                  LD      A,0A5H       ;SYNC BYTE VALUE INTO A
7E5E CD6402                CALL    264H         ;OUTPUT TO TAPE
7E61 C9                    RET
7E62 CDF801        UBER    CALL    1F8H         ;TURN OF CASSETTE
7E65 C9                    RET
                   ;
                   ;BEGIN TAPE READ ROUTINE
                   ;
7E66 3A907F        READ    LD      A,(FLAG)     ;FLAG VALUE INTO A
7E69 FE01                  CP      1            ;IF = 1
7E6B 280D                  JR      Z,READ1      ;  JUMP TO TAPE READ
7E6D FE02                  CP      2            ;IF = 2
7E6F 28F1                  JR      Z,UBER       ;TURN OFF CASSETTE & RET
7E71 3E01                  LD      A,1          ;SET FLAG VALUE
7E73 32907F                LD      (FLAG),A     ;  EQUAL TO 1
7E76 AF                    XOR     A            ;CLEAR
7E77 CD1202                CALL    212H         ;TURN ON CASSETTE
7E7A 2A8E7E        READ1   LD      HL,(VARPTR)  ;ADDR STRNG PNTR IN HL
7E7D 11907E                LD      DE,STRING    ;STRNG STORE ADDR IN DE
7E80 CD9602                CALL    296H         ;FIND SYNC BYTE
7E83 CD3502        READ2   CALL    235H         ;READ ONE BYTE
7E86 FE0D                  CP      0DH          ;CARRIAGE RETURN SIGNAL?
7E88 C8                    RET     Z            ;  YES, RETURN TO BASIC
7E89 12                    LD      (DE),A       ;PLACE CHAR INTO STORAGE
7E8A 34                    INC     (HL)         ;ADD ONE TO STRING LEN
7E8B 13                    INC     DE           ;BUMP STORGE POINTER
7E8C 18F5                  JR      READ2        ;GO FOR MORE
0002               VARPTR  DEFS    2
0100               STRING  DEFS    256
0001               FLAG    DEFS    1
0000                       END
00000 TOTAL ERRORS

READ2   7E83
READ1   7E7A
READ    7E66
WRITE3  7E56
WRITE2  7E48
UBER    7E62
WRITE1  7E40
FLAG    7F90
WRITE   7E29
TAB2    7E1E
GO      7E1A
TAB1    7E14
BKSPC   7E02
ZEROCK  7DFD
BUILD   7DEE
TAB     7E13
POLYBK  7DF8
ONEBK   7DFB
AGAIN   7DC9
REPEAT  7DC8
DISPLY  7E0D
STRING  7E90
VARPTR  7E8E
BEGIN   7DB5
```

Modification B. Changes in the "Editor/Formatter" to enable the LINE INPUT function and allow cassette I/O of delimited strings. This modification automatically loads machine-language Program 1 during the initialization process.

```
1 GOSUB4000:DELETE4000-4048
2 CLEAR4500:UN=1:Q=60:DIMF(Q),B(Q),L(Q),L$(Q)
•
•
25 L=L(M):P=1:I$=B$:PRINT"COMMAND ?";:AA%=VARPTR(I$):POKE32399,
   INT(AA%/256):POKE32398,256*(AA%/256-INT(AA%/256)):
   AA%=USR(0):GOSUB1000:IFR<0THEN40
•
•
124 I$="":PRINT"?>";:AA%=VARPTR(I$):POKE32399,INT(AA%/256):
    POKE32398,256*(AA%/256-INT(AA%/256)):AA%=USR(0):
    IFI$=""THEN2100
•
•
700 ' TAPE LOAD ROUTINE
```

```
710 PRINT"READY TAPE":INPUT"PRESS ENTER WHEN READY... ";R$
711 FORI=0TO59STEP10:INPUT#-1,L(I),L(I+1),L(I+2),L(I+3),L(I+4),
    L(I+5),L(I+6),L(I+7),L(I+8),L(I+9),F(I),F(I+1),F(I+2),
    F(I+3),F(I+4),F(I+5),F(I+6),F(I+7),F(I+8),F(I+9):NEXTI
712 INPUT#-1,UN
714 POKE16526,102:POKE16527,126:POKE32656,0:'CASSETTE ON SIGNAL
716 FORI=1TOUN-1
718 I$="":AA%=VARPTR(I$):POKEAA%+1,144:POKEAA%+2,126
720 POKE32398,256*(AA%/256-INT(AA%/256)):POKE32399,INT(AA%/256)
722 AA%=USR(0):L$(I)=I$:NEXTI
724 POKE32656,2:AA%=USR(0):'CASSETTE OFF SIGNAL
744 U=L(0):UN=F(0):M=U
746 FORI=1TON:B(F(I))=I:NEXT:B(U)=0:B(UN)=0:F(0)=0:B(0)=0
748 L(0)=0:POKE16526,181:POKE16527,125:'RESTORE LINE INPUT
749 GOTO25
750 ' TAPE SAVE ROUTINE
760 PRINT"READY TAPE":INPUT"PRESS ENTER WHEN READY... ";R$
765 L(0)=U:F(0)=UN
770 FORI=0TO59STEP10
771 PRINT#-1,L(I),L(I+1),L(I+2),L(I+3),L(I+4),L(I+5),L(I+6),
    L(I+7),L(I+8),L(I+9),F(I),F(I+1),F(I+2),F(I+3),F(I+4),
    F(I+5),F(I+6),F(I+7),F(I+8),F(I+9):NEXTI
772 PRINT#-1,UN
774 POKE16526,41:POKE16527,126:POKE32656,0:'CASSETTE ON
776 FORI=1TOUN-1
778 AA%=VARPTR(L$(I)):POKE32398,256*(AA%/256-INT(AA%/256))
780 POKE32399,INT(AA%/256):AA%=USR(0):NEXTI
782 POKE32656,2:AA%=USR(0):'CASSETTE OFF SIGNAL
790 POKE16526,181:POKE16527,125:'RESTORE LINE INPUT
791 GOTO25
•
•
4000 POKE16526,181:POKE16527,125:RESTORE
4010 FORI=32181TO32397:READA:POKEI,A:NEXT
4020 CLS:PRINT"TO RUN ** EDITOR/FORMATTER **"
4025 PRINT:PRINTTAB(5)"FIRST ENTER ==> DELETE 1"
4026 PRINTTAB(5)"THEN, ENTER ==> RUN":RETURN
4027 '
4028 ' OBJECT CODE FOR LINE INPUT FUNCTION
4029 '
4030 DATA 42,142,126,17,144,126,175,119,35,115,35,114,43,43,62
4031 DATA 14,205,13,126,213,205,43,0,183,40,250,209,254,1,200
4032 DATA 254,8,40,36,254,24,40,29,254,9,40,52,254,13,40
4033 DATA 42,254,32,56,223,205,238,125,32,218,24,13,18,205,13
4034 DATA 126,19,52,126,254,255,201,70,24,2,6,1,126,254,0,40
4035 DATA 196,62,8,205,13,126,27,53,16,247,24,185,213,205,51,0
4036 DATA 209,201,126,214,8,56,2,24,250,237,68,60,71,62,32,205
4037 DATA 238,125,40,214,16,247,24,157,58,144,127,254,1,40,16
4038 DATA 254,2,40,46,62,1,50,144,127,175,205,18,2,205,135,2
4040 DATA 42,142,126,70,35,94,35,86,26,19,205,100,2,16,249,62
4042 DATA 13,205,100,2,6,10,175,205,100,2,16,250,62,165,205
4044 DATA 100,2,201,205,248,1,201,58,144,127,254,1,40,13,254,2
4046 DATA 40,241,62,1,50,144,127,175,205,18,2,42,142,126,17
4048 DATA 144,126,205,150,2,205,53,2,254,13,200,18,52,19,24,245
```

get the message "NOT A COMMAND" if you were in line 25, or a blank text line if you were in 124, instead of the BREAK AT message. Simply tap the BREAK key twice in quick succession. The first tap gets you out of PGM 1; the second gets you out of execute mode before PGM 1 can be entered again.

Print Driver

As you may have noticed, not all of the protected memory is taken up by PGM 1. It ends at decimal address 32656 (7F90H). The reason for this is that I use the TRS-232 Printer Interface (a fine piece of hardware with excellent documentation and software provided, from Small Systems Hardware, Newbury Park CA). These remaining 100 plus bytes are, therefore, taken up with a print driver routine. You can put your keyboard debounce program, blinking cursor routine or whatever you wish in its place.

Conclusion

I hope you have as much pleasure in using the modified "Editor/Formatter" as I had in writing the mods and telling you about them. Also, now that you see how easy it is for BASIC to communicate with machine-language routines through the USR(0) statement, I hope you are encouraged to find additional applications for this versatile instruction. ■

Heath's H8 Cassette Interface Times Two

Transfer programs to and from a cassette and your H8 at twice the speed.

Ronald Baker
6102 Crockett Dr.
Louisville, KY 40258

This article will show how you can transfer programs to and from a cassette recorder to your H8 at twice the speed with a simple modification. In experimenting with my H8 I am constantly transferring data to and from the cassette recorder. One of the things that sold me on the H8 system was the 1200-baud rate of their cassette interface. After using the system for a few months even this speed seemed like it took forever to load or dump a program.

In building the cassette interface board I remembered that I had a choice of two speeds: 300 or 1200 baud. If I had a choice of two speeds, why not another choice of even a higher speed?

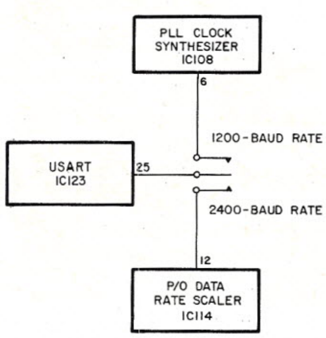


Fig. 1. The switching arrangement between circuits.

Upon reviewing the schematic, I noticed that Heath uses a nifty circuit with a phase-lock loop that automatically compensates for tape-speed variations. This circuit could be modified for a faster rate, but not too easily. I figured that there had to be a better way. Furthermore, I didn't want to permanently wire the system for a higher baud rate.

All of Heath's program tapes are recorded at a 1200-baud rate, which means I couldn't use it if something new from Heath came along in the future. Since I wanted to operate at a faster rate than 1200 baud, I tried to design a simple modification that would accommodate both Heath's 1200-baud-rate cassettes and my own system to operate at 2400 baud.

Theory

The construction theory is simple. With one exception, the 1200-baud-rate wiring of the board stays the same. The output of the phase-lock-loop clock synthesizer now goes to the USART (universal synchronous/asynchronous receiver/transmitter) IC123, pin 25, through a single-pole double-throw switch as shown in Fig. 1.

This position on the switch selects the 1200-baud rate. The second position on the switch disconnects IC123 from the clock synthesizer and connects to pin 12 of IC114, the data rate scaler, which is also the 2400-baud-rate clock output.

Thus, when the board is operating at 2400-baud, the clock synthesizer is not used. Instead the clock pulse for the USART comes from the crystal-controlled baud rate generator, IC112 through IC116.

At this point, the disadvantage of disconnecting the clock synthesizer should be obvious. This circuit is what gives you a constant clock frequency with any variation in tape speed. I'll admit that at first I had second thoughts about how well it would perform. There was only one way to find out and that was to try it and see.

The results were quite surprising. I have been using this system at 2400 baud for about six months now without a sign of any problems. It works as well as the 1200-baud rate, but now it's twice as fast.

At this rate of speed your recorder is very important. Battery-operated recorders are out. Tape-speed variations are too great with the discharge of the batteries. The results will be a checksum error. The recorder I use is a good, inexpensive ($100 price class) Radio Shack tape deck that runs on 110 V ac. If you use a recorder that has a wow and flutter specification of 0.25 percent or less, it should perform well without any difficulties.

Construction Is Easy

The actual construction of the additional circuit takes less than one hour. The only part needed is a single-pole double-throw subminiature slide switch, which most parts suppliers can provide. Since 99 percent of my operation is at the 2400-baud rate, I mounted the switch directly on the cassette interface board. This makes the switch accessible if I need to switch to the 1200-baud rate, and out of sight for most of my operating. Another advantage of mounting it on the board is that it obviates drilling a hole in the front panel, thus destroying the appearance of the H8.

As shown in the photograph, the switch is mounted in the small vacant area between IC108 and IC113. Note that on the backside of the board this

same area is free of printed circuit. The switch is mounted vertically by drilling three small holes. Size and distance of the holes will vary with the type of switch used.

The end result is that the switch is flush with the board; the switch contacts are on the backside of the board. The switch is held in position on the board with a small drop of epoxy on either side of the switch.

After the switch is mounted, the second step is to cut and remove the foil between the USART IC123, pin 25, and IC108, pin 6. Important: make sure when you cut the foil that the foil connecting IC108, pin 6, to IC107, pin 14, is left connected.

The only thing left now is the wiring.

1. Connect a wire from IC123, pin 25, to the middle contact of the switch.

2. Connect a wire from the junction of IC108, pin 6, and IC107, pin 14, to the top contact of the switch.

3. Connect a wire from the bottom of the switch to IC114, pin 12. This is the clock reference for the 2400-baud rate.

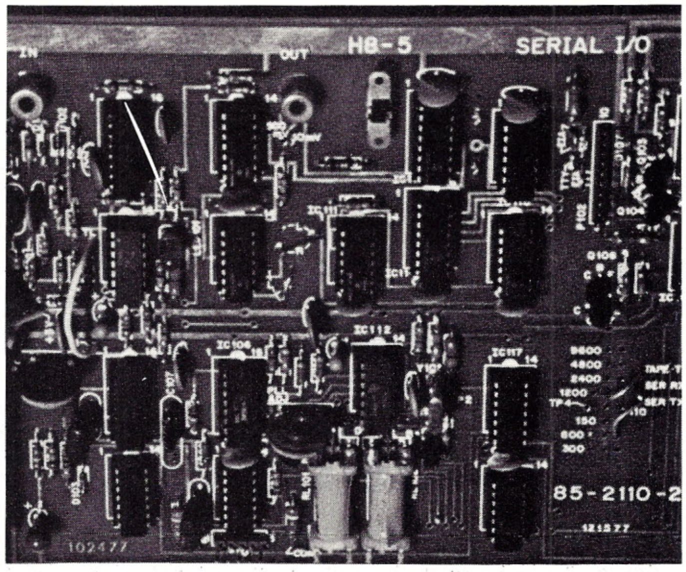
4. Connect a wire from the TAPE TX (as marked on the board) to the 2400 position (as marked on the board). The first three wiring steps convert the tape receive circuit. When the switch is up the transfer rate is 1200 baud. The rate is 2400 baud when the switch is down. The fourth wiring step converts the tape transmit circuit to the 2400-baud.

Operation

Conversion of your tapes to the 2400-baud rate is simple. Insert a tape previously recorded at 1200 baud into your recorder. Set the switch to the 1200-baud rate. Press the load button on the H8 to load normally. After the program has been loaded, insert a blank cassette into your recorder and press the dump button on the H8 to make a new copy at the 2400-baud rate. Important: use a blank cassette to make a new copy.

Never use the old program tape until a good copy has been made at the new rate. If something went wrong in dumping the program, the original program would be lost. For all programs written in machine language, use the procedure just described.

Use the same procedure to load programs written in BASIC; then use the load and dump commands of BASIC in the same manner. After you convert all your program tapes to the new rate, you can leave the switch in the 2400-baud-rate position for normal use.



The switch is mounted between IC108 and IC113 . . . just below the letters H8-5 printed across the top of the board.

Conclusion

Probably the most important part of the system's operation is your tape recorder. Inexpensive battery-operated recorders do not consistently give you reliable service. Use a good-quality tape recorder that operates from 110 V ac.

In selecting a recorder, also keep in mind the frequency response of the unit. When you operate at the 2400-baud rate, the increase in frequency is important.

For the same reason, use a good-quality recording tape. Keep the recording heads clean, and perform some maintenance on your tape recorder occasionally since this might prevent a lot of trouble later.

The recording level is not critical. It should be about the same at the new rate as at the old rate. If adjustments are necessary, consult the operating manual.

Since the tape transmit circuit is permanently wired to the 2400-baud rate, whenever you dump a program to your recorder it will be recorded at 2400 baud. This means you can load a program at either 1200 or 2400 baud but you can only dump a program at 2400 baud.

This provides a useful, simple and relatively inexpensive method of transferring programs to and from a Heath H8 at twice the normal speed. ■

PET Pen

Part 2 of this 3-parter adds a light pen to last month's PET I/O port expander.

William F. Pytlik
9012 Maritime Court
Springfield, VA 22153

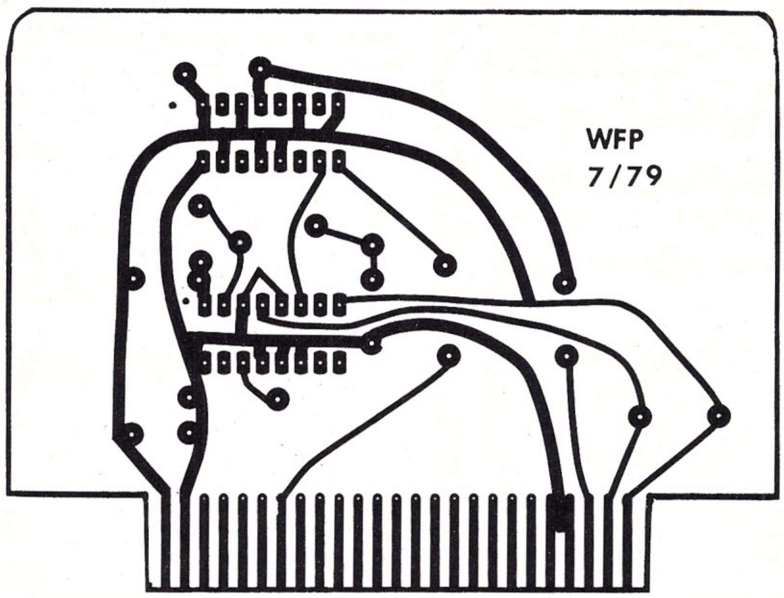


Fig. 2. PC board (full size).

Providing the PET with "eyesight" greatly enhances its capabilities. To take full advantage of the PET's interactive capability, you can add a light pen, which will provide some measure of "eyesight." Darlington-connected phototransistors make implementation of light pens easy. Using the I/O port "Expander" described in last month's article, you can add a simple plug-in printed circuit (PC) card to provide all the circuitry required to interface the light pen with the PET. This article describes the hardware as well as the Shopping List software routine to demonstrate use of the light pen.

Circuit

Fig. 1 shows the schematic of the light pen circuit. The phototransistor, when properly biased and exposed to the light of a blinking cursor, provides an output pulse. This pulse is applied to a one-shot (IC1), which squares and lengthens the pulse. The output of IC1 is then buffered by IC2 and transferred to the I/O port via the I/O port Expander. Position of switch S1 (Device Select 2 on the Expander front panel) determines whether the signal is passed to the I/O port.

If +5 V dc is applied to pin 4 of IC2, the output goes to a high impedance state—essentially isolating the light pen module from the I/O port. If pin 4 of IC2 is grounded, the pulse is transferred to bit 8 (PA7) of the PET I/O port. At this point, you must use software if you want to utilize the light pen.

Construction

The plug-in module is constructed using a 3 × 4 inch PC board, shown in Fig. 2. Although I used photographic PC board

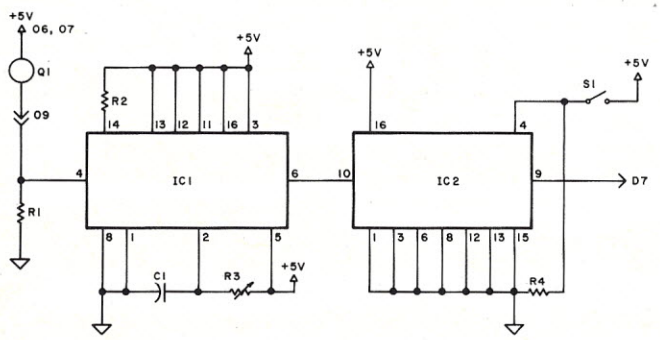


Fig. 1. Light pen circuitry.

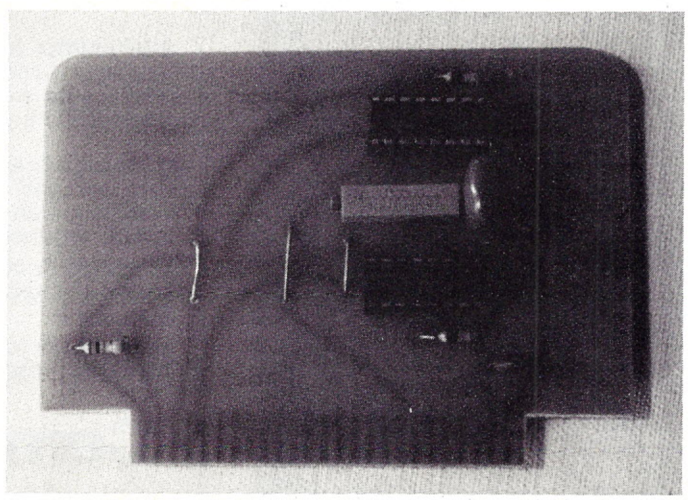


Photo 1. Completed PC board.

construction techniques, you may use either Datak dry transfer etch patterns or a .1 inch grid PC board and a resist pen. All components are mounted and soldered in place using Fig. 1 as a guide. Photo 1 shows the completed PC board.

The light pen consists of an appropriate housing (I used a refillable felt-tip pen barrel), two-conductor wire and a DIN male plug. Fig. 3 presents a sketch of the light pen. Photos 2 and 3 show the components and the complete light pen, respectively.

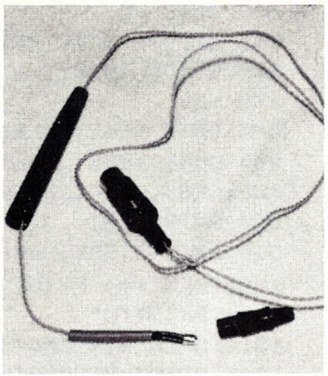


Photo 2. Light pen components.

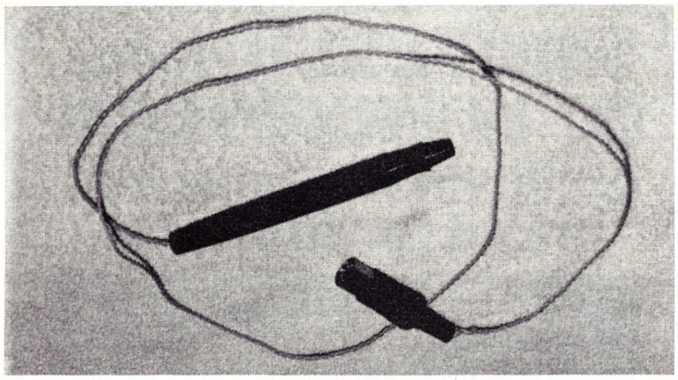


Photo 3. Assembled light pen.

Software

I've written the Shopping List program to demonstrate just one of the many uses of the light pen. The program, in conjunction with the light pen, provides a method by which you can obtain a printout of a number of items from a long list by simply pointing the light pen toward desired items.

Lines 65 to 150 read the category and a page worth of items. For example, the category might be vegetables and the items lettuce or peas. The data is appropriately formatted and displayed on the screen. At the bottom of the screen, the program prints NEXT PAGE, allowing the user to go to additional data in the list at any time. Subroutine 2000 writes a cursor, which travels from the first item to the bottom of the page. (Note: R3 and screen brightness should be adjusted for best operation; I adjusted R3 to approximately 300k.)

In addition to writing a cursor, the subroutine also reads the I/O port. If the pen is placed in the path of the cursor next to a desired item, the subroutine will save this item in memory. If the pen is placed next to NEXT PAGE, the program will provide the next page of items. When all items have been displayed, the program goes to the print subroutine at line 1000. (Note: Line 1010 and 1150 are written for my Axiom 801P printer. If you have a different printer, you may have to change these lines. If you do not have a printer, adjust this subroutine so that only 20 of the items are displayed at a time.)

The list of items to choose from is entered starting at line 500. I used only a few items in this sample to conserve space. Note that item END specifies the end of a category, while STOP specifies the end of the list.

My wife regularly uses the light pen to obtain a shopping list in a matter of minutes. Another application is administration of multiple-choice test questions. If you write the cursor across the entire screen (remember: before writing the cursor, you must save the affected screen data element and restore it after the cursor is moved), you can use the light pen to draw simple designs.■

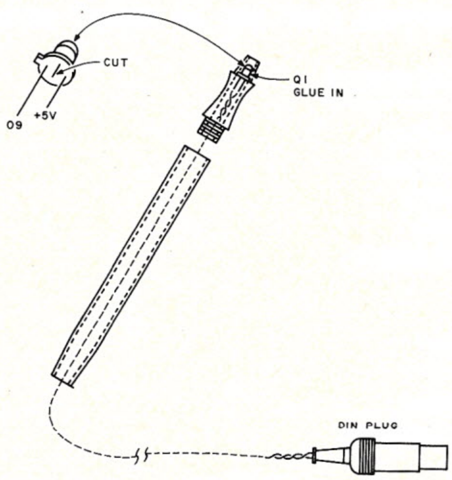


Fig. 3. Light pen construction.

IC1	–	CD4528 CMOS dual one-shot
IC2	–	CD4502 CMOS hex buffer
R1	–	10k ¼ Watt
R2,R4	–	100k ¼ Watt
R3	–	1 meg trimpot
C1	–	.1 μF
C2	–	.01 μF (use between 5 V and ground)
Q1	–	GE L14F-1 Darlington-connected phototransistor
Misc.	–	Hardware, wire, IC sockets, etc.
Parts list		

Parts list.

Program listing.

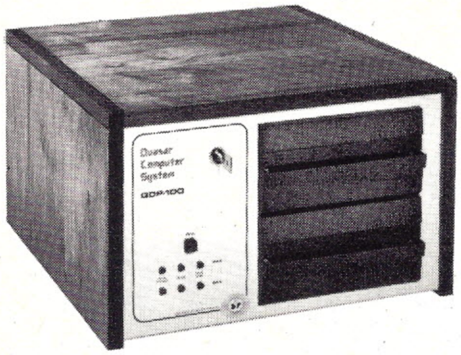
```
 10 REM *** SHOPPING LIST ***
 20 REM **OPEN I/O PORT FOR INPUT**
 30 INPUT "ENTER TODAYS DATE";TD$
 40 DIM CA$(20),ITEM$(20,20),OUT$(20,20)
 50 A=1:A1=1
 60 X=0:Y=0
 65 READ CA$(A)
 70 PRINT"CATAGORY--";CA$(A):PRINT:PRIN
T
 80 REM **DISPLAY ITEMS**
 90 FOR I=1 TO 10
 100 READ ITEM$(A,I)
 105 II=I
 110 IF ITEM$(A,I)="END" THEN Y=1:GOTO 1
50
 120 IF ITEM$(A,I)="STOP" THEN X=1:GOTO
150
 130 PRINT"";ITEM$(A,I):PRINT
 140 NEXT I
 150 PRINT"                NEXT
PAGE"
 160 REM **GO TO LIGHT PEN ROUTINE**
 170 GOSUB 2000
 180 IF Y=1 THEN A=A+1:GOTO 60
 190 IF X=1 THEN 210
```

```
200 GOTO 70
210 REM **GO TO PRINT ROUTINEE**
220 GOSUB 1000
230 END
500 DATA VEGETABLES
510 DATA TOMATO,LETTUCE,BEETS,CARROTS,B
EANS,PEAS,CORN,CELERY,CUCUMBER,END
520 DATA MEAT
530 DATA STEAK,HAMBURGER,CHICKEN,STOP
1000 REM *** PRINT ROUTINE ***
1001 PRINT"♥"
1010 OPEN 4,4:CMD4:PRINT CHR$(8)
1015 PRINT "SHOPPING LIST FOR ";TD$
1020 FOR I=1 TO A
1030 PRINT
1040 PRINT CA$(I)
1050 FOR C=1 TO LEN(CA$(I))
1060 PRINT"¯";
1070 NEXT C
1080 PRINT
1090 FOR J=1 TO A1
1100 IF OUT$(I,J)="END" OR OUT$(I,J)="S
TOP" THEN 1120
1105 IF OUT$(I,J)=OUT$(I,J-1) THEN 1120
1110 PRINT OUT$(I,J)
1120 NEXT J
1130 NEXT I
1140 PRINT
1150 PRINT#4:CLOSE4
1160 RETURN
2000 REM *** LIGHT PEN ROUTINE ***
2010 PRINT"⊠"
2020 FOR I=1 TO 11
2030 PRINT" █████ ███";
2040 IF PEEK(59471)<250 AND I>10 THEN R
ETURN
2050 IF PEEK(59471)<250 AND I<II+1 THEN
OUT$(A,A1)=ITEM$(A,I):A1=A1+1
2060 REM **SLOW CURSOR DOWN**
2070 FOR T=1 TO 1:NEXT T
2080 NEXT I
2090 PRINT" "
2100 GOTO 2010
```

```
SHOPPING LIST FOR 12 AUGUST 1979

VEGETABLES
----------

TOMATO
LETTUCE
BEETS
CORN
CUCUMBER


MEAT
----

STEAK
HAMBURGER
```

Sample run.

You can have it all . . . .



Z-80
OR
Z-8000

QDP – 100
Z-80 Computer

- Z-80 4 Mhz CPU
- 64 K Memory
- Two 8″ Double Sided Double Density Floppies
- Smart CRT Terminal
- Four Ports - 2 Serial - 2 Parallel
- Disk Controller - Double Sided Double Density - up to 4 M.B.
- Monitor in Prom
- CP/M 2.2 DOS
- S-100 (IEEE) Motherboard
- Basic Language Included
- Real Time Clock
- Upgradable to QDP-8100

$4995

UNIX for Z-8000 Available soon

Z-8000 IEEE 100 CPU Board
$495.00

QDP-8100
Z-8000 Computer

- Z-8000 4 Mhz CPU IEEE-100
- 64 K Memory
- Two 8″ Double Sided Double Density Floppies
- Smart CRT Terminal
- Three Ports - 2 Serial - 1 Parallel
- Disk Controller - Double Sided Double Density - up to 4 M.B.
- CP/M 2.2 DOS
- Z-80 Software Emulator
- I-EEE 100 Motherboard
- Basic Language Included
- Z-8000 Pascal Available

$6395

SOFTWARE FOR OUR SYSTEMS

- Professional accounting system G/L A/P A/R P/R Inv
  Fully integrated and easy to use................$1495.00
- Osborn Accounting Package G/L A/R A/P P/R
  Small business accounting package..............$449.00
- Data Base Management System - Selector III.......$295.00
  Selector IV.......$525.00
- Data Star..$350.00
- Word Processing - Wordstar.......................$445.00

Z-80 Emulator - allows Z-8000 to run Z-80/8080 Programs..**$295.00**

Z8000 PASCAL - the first Z-8000 native code Pascal **$595.00**

4 Mhz 64K Dynamic RAM
16K - $250.00 32K - $350.00 48K - $450.00 64K - $549.00

TELETEK DBL. DENSITY, DBL. SIDED
Disk Controller Board...............................$395.00

SINGLE BOARD COMPUTER
4 Mhz Z-80, DBL density disk controller, 2716 prom burner 2 parallel & 2 serial ports, real time clock, Bios for CP/M 2.2, monitor in prom. This is absolutely the best board we have ever seen. It plus the 64K memory board gives you a complete system.
One year parts & labor guarantee. (The Best)........$795.00

QUASAR FLOPPY SYSTEM
- Two MFE DBL sided drives • Cable • Case & Power Supply assembled and tested Wood cabinet $1895.00

QUASAR 2 MEG FLOPPY
- 2 MFE double sided drives
- Teletek disk controller board
- Power supply & cable
- Wood cabinet
- CP/M version 2.2 & bios
- Assembled & tested $2295.00

MFE Double Sided - Double Density 8″ Floppy Disk Drives. (the best) $650.00
Using the Teletek Controller under CP/M, THIS DRIVE WILL GIVE YOU ALMOST ONE MEGABYTE PER DISK DRIVE.
Power supply for above $110.00

PAPER TIGER
Includes Graphics $949.00
Cable for TRS-80 ......$39.00
Call for Apple

TI - 820
Serial Printer -
Full package options...$1995.00

Dealer Inquiries Invited, Hours:9-5:30 M-F
Specifications Subject To Change
UnixTM - Bell Lab 30 Day ARO CP/MTM - Digital Research

Checks, money orders accepted
Add $2.50 freight charges on orders under 10 lbs. Over 10 lbs. F.O.B. Cleveland

QUASAR DATA PRODUCTS ✓295
25151 Mitchell Dr., No.Olmsted, Ohio 44070 (216)779-9387

64K Memory For the Heathkit H8

This single card dynamic RAM plugs directly into the H8.

Myron J. Seibold
PO Box 5131
Santa Ana, CA 92704

This article describes a single card, 64K byte, random-access memory designed for the Heathkit H8 computer. The memory is packaged on a 6 inch (15.2 cm) × 12 inch (30.4 cm) board using standard Heath mounting hardware and plugs directly into the H8. The memory bytes are 8 bits in length. The total capacity of the memory, then, is 64K × 8 bits. Of this, only 56K can be used, because Heath reserves the first 8K of memory addresses for other uses. This memory may be addressed from 8K to 64K bytes. It will not respond when addressed in the 0 to 8K bytes range.

This is a general-purpose memory design, which should be adaptable with little modification to almost any small computer. I will show how this memory interfaces with the H8 bus, and, with this example, general memory interfacing requirements will be demonstrated.

A single-card memory is made possible by using the 16K × 1 bit dynamic memory integrated circuit chip. This chip—the 4116—became available over two years ago and is now the industry-standard random-access memory device. It can now be obtained from many manufacturers at reasonable prices and should soon be finding its way into many home computer designs.

Memory Chips

Semiconductor memories began to compete successfully with magnetic core memories with the introduction of the 4K × 1 bit metal oxide semiconductor (MOS) memory chip. 4K memory chips became available about 5 years ago and were introduced in a wide variety of styles—16-, 18- and 22-pin parts. Some used static storage techniques, while others used dynamic techniques. There was little compatibility between manufacturers of similar parts.

Static memory chips store information in a flip-flop circuit, usually consisting of six transistor elements. Stored information is nonvolatile as long as power is applied. Static memory chips require little additional support circuitry and are easy to use. Memory expertise is generally not required to design with static memory chips.

Dynamic memory chips store information as a charge on a capacitor. This capacitor is usually connected to the other memory circuits on the chip with a single transistor switch. A separate capacitor and switch are required for each bit of information stored on the chip. The charge stored on the capacitor gradually leaks away and must be periodically restored. This operation is called refresh. Dynamic memory chips require considerable additional support circuitry. Good printed circuit board layout practices are essential.

Dynamic memory chips have many advantages over static chips. They are less expensive per bit of stored information, store more information per chip and use less power. They provide significant savings in computer chassis space.

One 4K dynamic memory chip (type 4096) was introduced in a 16-pin package that had six address inputs. The 12 memory address bits required to address 4K were multiplexed into the memory chip in two steps, using these six inputs. This was considered a very radical technique, and it was predicted that this chip could not compete with more conventional 4K memory chips.

Not only was this chip very successful, it provided the technique required to build the next generation memory device: the 16K × 1 bit dynamic memory chip. All 16K memory chips are alike, and chips from all manufacturers may be used interchangeably. This is also a 16-pin part, with seven multiplexed address inputs. Static 16K memory chips are not available.

At this time, 64K × 1 bit dynamic memory chips (type 4164) have been announced. These parts also use 16 pins and have eight multiplexed address inputs. These new 64K chips use a single power supply voltage, +5 volts. The older 4K and 16K chips use three power supply voltages: −5 volts, +5 volts and +12 volts.

Using the 16K Dynamic Memory Chip

The 4116, 16K × 1 bit memory chip package has 16 pins, as previously noted. Seven of these pins are address inputs, three are power supply inputs, one is for data input, one is for data output and one is ground. There are also three timing strobes: RAS (Row Address Select), CAS (Column Address Select) and WRT (Write). All inputs and outputs (address, data and timing strobes) are TTL compatible. They may be connected to standard TTL integrated circuit inputs and outputs.

The three timing strobes are normally held at a high level and are asserted when driven to a low level. The memory cycle starts when RAS is asserted. The seven address inputs must be applied before this time. These addresses are latched into the memory chip when RAS is asserted. A short time after RAS is asserted, seven new addresses are applied to the memory chip address inputs. (A total of 14 address bits are required to address 16K of memory.) CAS is then asserted, latching these additional seven address bits into the memory chip. WRT should

Memory chip	Access time
4116-2	150 NS
4116-3	200 NS
4116-4	250 NS

Table 1. Memory chip access times.

remain high for a memory read cycle. Valid memory data will appear at the memory chip data output after a certain time following the assertion of RAS.

This time (memory chip access time) is specified according to the grade of memory chip used. Three grades are available; their respective access times are given in Table 1. The fastest chip, the 4116-2, is the most expensive. All three grades are made using the same process and are screened into these categories by testing. Memory data will remain valid at the memory chip output terminal until the end of the memory cycle, when RAS and CAS are released.

WRT is asserted for a memory write cycle. Input data is latched into the memory chip upon the assertion of either CAS or WRT, whichever is asserted later. If the input data is changed after this time, it will not be stored in the memory chip.

I included this information concerning the 4116 memory chip to aid in understanding the operation of the memory. It is not sufficient for design purposes or to make design changes. Further information may be obtained from the manufacturer's data specification sheet for this part.

Refreshing the Memory Chips

The entire memory must be refreshed once every two milliseconds. A memory refresh cycle differs from a normal memory cycle in that only RAS is asserted. The memory must be refreshed for all row addresses. There are seven row address inputs; therefore, the memory must be refreshed 128 times, every two milliseconds. The refresh cycles are normally evenly spaced 16 microseconds apart. This will provide 128 refresh cycles in 2 milliseconds. A refresh cycle typically has a duration of one-half microsecond. The memory will, therefore, be performing refresh cycles less than 5 percent of the time.

Memory refresh cycles potentially interfere with computer memory cycles. Refresh cycles must be synchronized with the computer to prevent interference. The operating cycles of most computers can be extended by the assertion of a "stall" signal to accommodate memories with excessive access times. Use can be made of this provision when the memory is performing a refresh cycle.

The start of a computer memory cycle will be delayed until completion of the refresh cycle, and the computer stall signal, meanwhile, will be asserted until the memory can accept or provide computer data. The Heathkit H8 bus has a signal line called RDYIN on connector pin 20, which may be used to extend the operating cycle of the computer.

Frequently, memory refresh cycles can be timed so that the computer does not have to wait for the completion of a memory refresh cycle. In this case, the computer stall signal need not be asserted. Refresh is then said to be transparent. The stall signal may be used once on start-up and occasionally when the computer is using the memory at a low repetition rate. The stall signal will not be asserted when the computer is using the memory continuously.

This memory design features transparent refresh. There is sufficient time between computer memory cycles to follow a computer memory cycle with a memory refresh cycle. Refresh cycles will be generated in this case only when required. The memory should not be refreshed excessively, as the power requirements for refresh are significant.

Interfacing the Memory to the Computer Bus

The Heathkit H8 bus operates the memory with a minimum of signal lines. There are 16 address lines to address up to 64K bytes of memory. There are eight bidirectional data lines, which can either supply data to the memory or receive data from the memory in 8-bit bytes. There is a memory read input and a memory write input. Provision for a memory output to stall the computer (RDYIN) is also available on the bus. There are three power supply voltages, as well as two ground connections. A total of 32 connections is made to the memory from the computer bus.

The H8 memory cycle has a duration of 1500 nanoseconds (see the memory timing relationships shown in Fig. 1). The memory read command pulse width is 750 nanoseconds; the memory write command pulse width is 1150 nanoseconds. A memory cycle will be either a read cycle or a write cycle, depending upon which memory command is used. A memory cycle nominally begins when a memory address is applied to the address lines. This occurs 200 nanoseconds prior to the assertion of either a read or a write command. Both the read and write commands are positive pulses on the bus.

Memory data is expected on the bus 300 nanoseconds after the assertion of a memory read command. Similarly, data to be written into the memory is available on the bus 300 nanoseconds after the assertion of a memory write command. Address and data information remain valid on the bus until the release of the memory read or write commands.

The memory access time requirement, measured from the assertion of the memory addresses on the bus, is 500 nanoseconds. The 8K static memory used by Heath (H8-1) has a specified access time of 450 nanoseconds. However, the static memory cycle starts when the memory addresses are asserted. The dynamic memory cannot start at this time and must wait until a memory read or write command is applied. Accordingly, 200 nanoseconds are lost from the memory access time. This is the primary reason for using the faster "dash 3" (4116-3) memory chips. With these, the memory module access time measures 500 nanoseconds and can be set faster if required.

Memory read data is applied to the bus through a latching bus driver. This driver has a high-impedance output, unless it is applying data to the bus. Read data remains latched on the bus

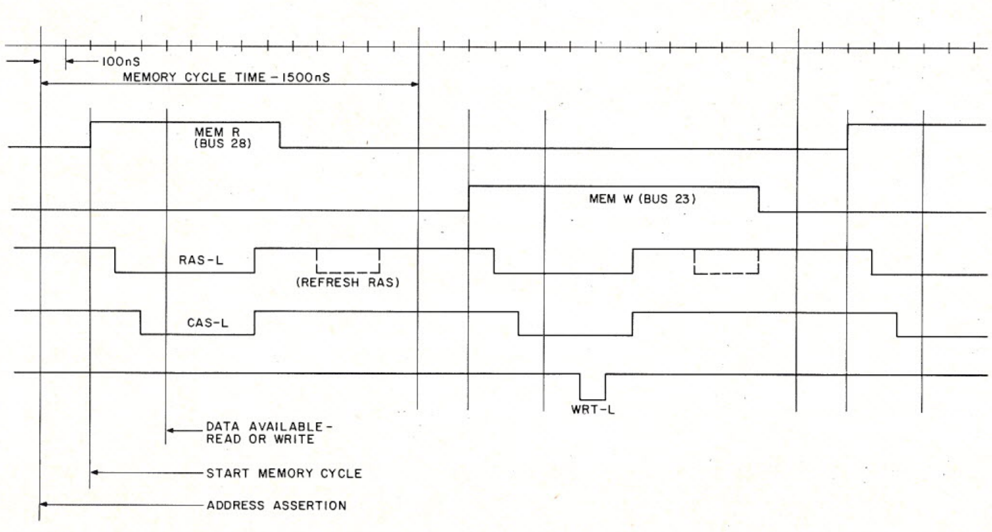


Fig. 1. Memory timing relationships.

until the release of the memory read command. The memory itself is free for refreshing immediately after data is latched on the bus.

Data to be written into the memory may not be available until 300 nanoseconds after the assertion of the memory write command. Data must be latched into the memory chips, using the WRT command, after this time.

As previously stated, the memory cycles are started by the assertion of the memory read or write commands. The duration of these commands has no affect on memory operation. The memory cycle finishes as soon as possible, followed by a short pause. The memory is then ready to begin a refresh cycle. There is time for a refresh cycle and another pause before the next memory cycle will be requested. Synchronized in this way, memory refresh cycles do not delay execution of the computer memory cycles.

The extended duration (1150 nanoseconds) of the write cycle command is a potential problem. This enables the computer to write late data into a static memory. The dynamic memory can be easily modified to write late data, but at the expense of transparent refresh. Since no problem exists at present, I decided not to worry about late data entry. A problem could conceivably exist in the future with certain types of interfaces or system configurations. In this case, the memory cycle time must be extended and application of WRT delayed accordingly. The provision for transparent refresh must then be disabled.

To adapt this memory to a computer other than the Heathkit H8, the computer bus requirements will have to be determined and appropriate changes made in the memory interface circuits and possibly in some of the timing circuits. The best way to examine the computer bus signals is to probe the bus terminals using an oscilloscope while running the computer using a short program loop entered in machine code.

The use of these test loops is invaluable, as known operating conditions can be established on the bus. These loops should be made as short as possible. Different loops can be used to provide different bus conditions, for example, (1) store accumulator in memory (write into memory) and return or (2) load accumulator from memory (read from memory) and return. Front-panel monitor routines should be disabled when running these loops, if possible. The bus signals can be spread out in intelligible patterns by synchronizing the oscilloscope on an appropriate higher-order address bit. This synchronizing address change can be written into the beginning of the test routine.

The Heath H8 bus operation is conventional, and little modification should be required to adapt this memory for use with another computer. As noted, it will probably be necessary to change a few of the input circuits to convert this memory to another computer bus. The memory timing is essentially independent of the computer, and little modification of the timing should be required in this case. I modeled this memory after one of my best commercial designs, which I readily adapted to the H8 bus.

If another computer does not provide sufficient time between memory cycles for transparent refreshing, this circuit in the memory should be disabled by connecting U56-3 (see Fig. 2) to a +5 volt bias resistor (or to ground) instead of to U69-12. When this is done, all refreshing will be asynchronous and the computer stall signal will be used to prevent interference. The increase in computer oper-

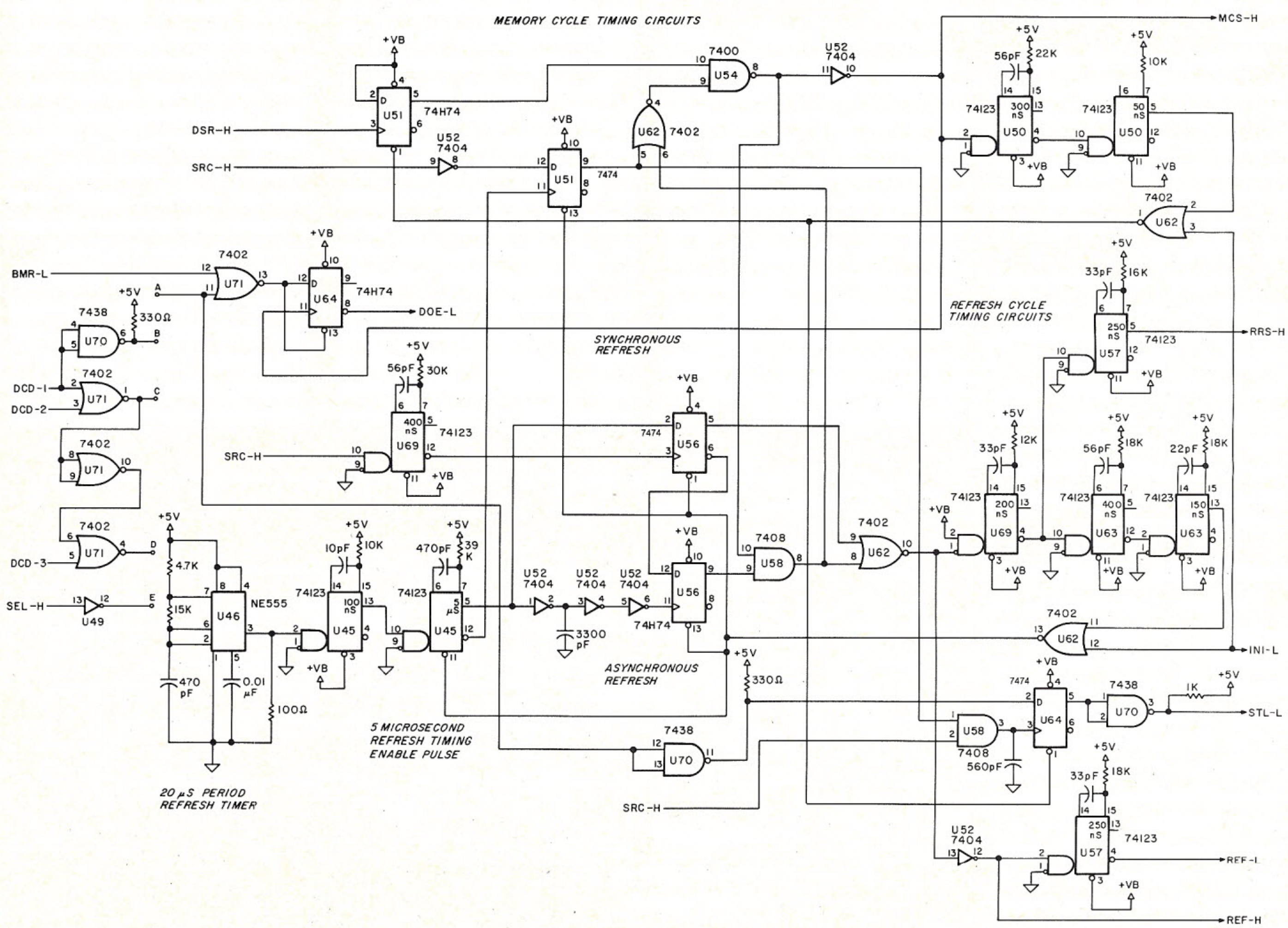


Fig. 2. Memory timing circuits and refresh synchronization.

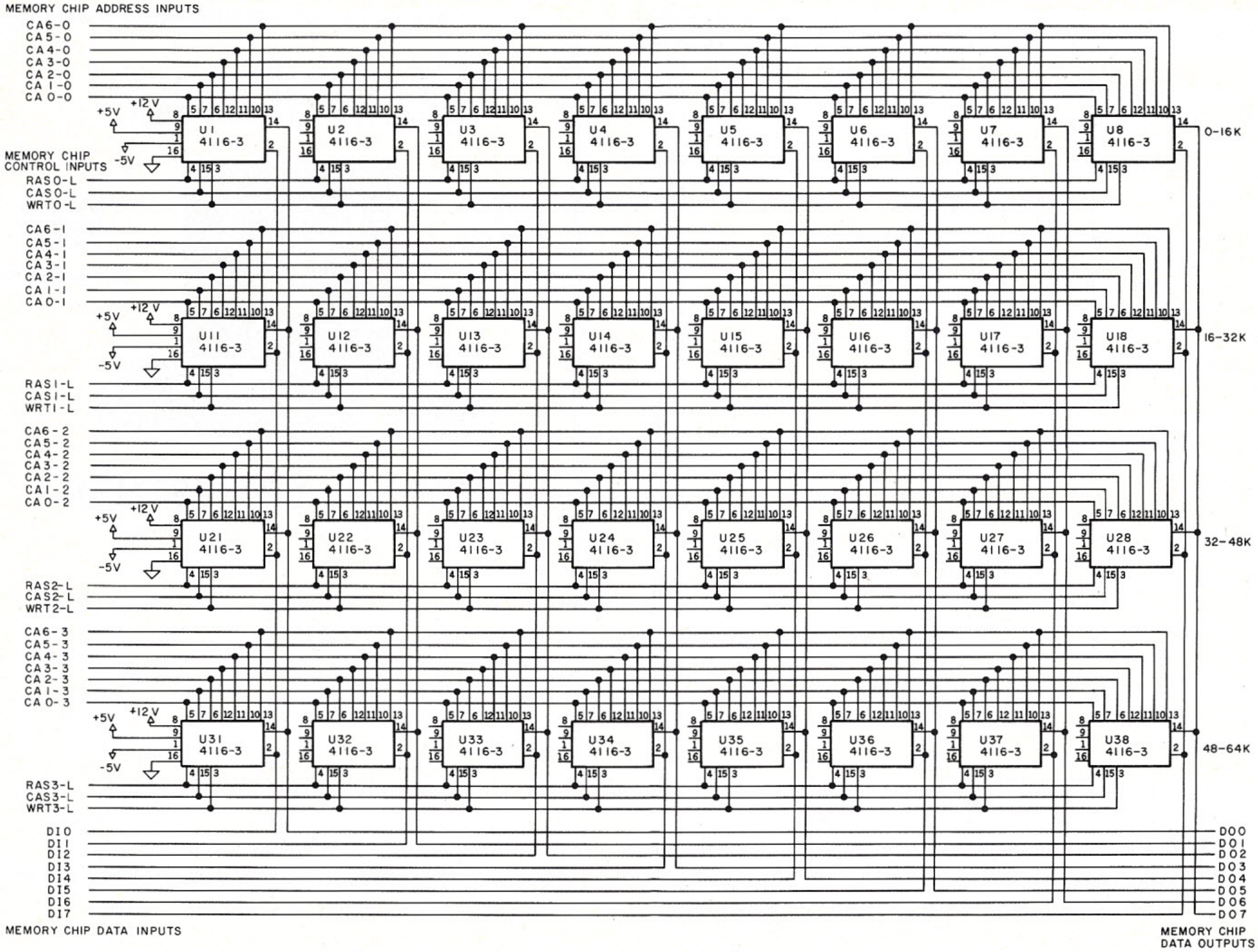


Fig. 3. Memory data storage device array (64K × 8 bits).

ating time required for asynchronous refresh usually will not be noticeable.

Address Range Selection

You can build this memory in a smaller size version having less memory capacity by simply omitting one or more rows of memory chips. These chips can later be added, one row at a time, to increase memory capacity. A smaller version of this memory is fully compatible with any existing memory now being used in the H8 computer. This new memory always runs at the lower memory addresses. Existing H8 memory must be rejumpered to run above this memory.

There are four address selection jumpers on this memory. They select one of the following address ranges: 8K to 24K, 8K to 40K, 8K to 56K and 8K to 64K. These address ranges correspond to memory sizes of 16K, 32K, 48K and 56K, respectively. The address selection jumper must be installed for the appropriate case. See Table 2 for address selection jumper installation. Additional memories used in the H8 computer must be jumpered continuously above the range selected for this particular memory.

I incorporated this provision for address selection after the memory was designed and built. There is no need for this feature with a full 64K memory. Integrated circuit number 71 is not shown on the memory module layout drawing (Fig. 3). It will have to be squeezed in somewhere. Perhaps it will fit between U68 and U69. Use of a 16K memory chip (4116) results in a 16K minimum increase in memory capacity. The address range increments in 16K steps (from a minimum 16K capacity), accordingly.

Install Jumper Wire	Memory Address Selection Range
A to B	8K to 24K
A to C	8K to 40K
A to D	8K to 56K
A to E	8K to 64K

Install a Single Jumper Wire as Indicated Above

Table 2. Jumper wiring for memory address range selection.

Memory Module Circuit Description

The memory continuously generates refresh cycles whether the computer is using the memory or not. Much of the memory circuitry can be checked and the timing adjusted simply by working with the refresh signals. This is important because some initial adjustment may be required before the computer will work with the memory.

Refresh cycles are timed by oscillator U46 (see Fig. 2), whose period is approximately 16 microseconds. A 100 nanosecond pulse is generated every 16 microseconds by applying the output of this timer to one-shot U45. A 5 microsecond refresh enable timing slot is generated when the 100 nanosecond pulse is applied to the other half of U45.

If a memory cycle occurs during the 5 microsecond refresh enable interval, a refresh cycle will be initiated immediately following the memory cycle. In this case, the output at U56-5 will start the refresh cycle. If there is no memory cycle, the termination of the 5 microsecond timing interval will initiate the refresh cycle. In this case, the output at U56-9 will start the refresh cycle.

In either case, the refresh cycle will start when U62-10 is asserted. A delay set by U69-4 will

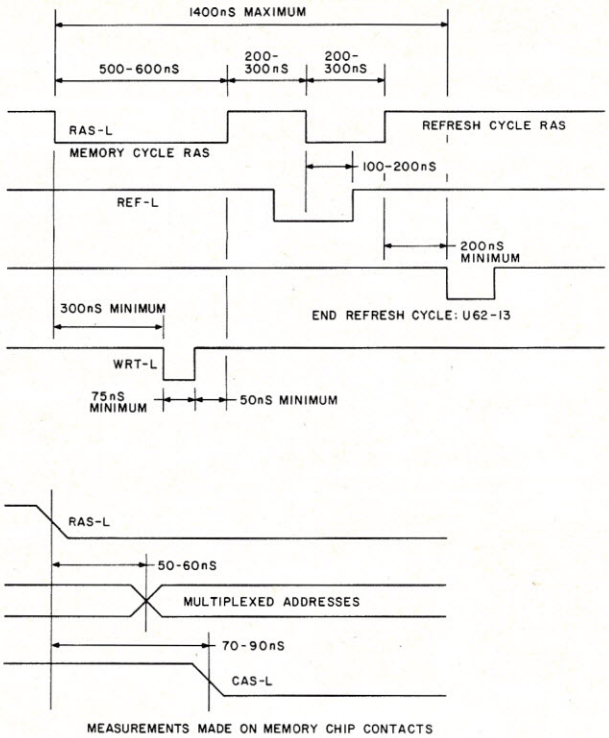


Fig. 4. Detailed timing specification.

occur before the application of refresh RAS to the memory chips. A delay is required between successive applications of RAS to the memory chips, between the memory and refresh cycles. During this delay, the address multiplexer applies the outputs of the refresh counter to the memory chip address inputs. The refresh counter is advanced with each refresh cycle. When refresh RAS is removed from the memory chips, there is another delay, after which the refresh cycle is terminated. The memory is now ready to begin a computer memory cycle.

A computer memory cycle is initiated when the input at U51-3 goes high. This will cause the output at U51-5 to go high, provided a valid memory address has been received. If the address is in the range 0 to 8K, the input at U51-2 will be low and the output at U51-5 will not change. If a refresh cycle is in progress, the input at U54-9 will be low and a memory cycle will not begin until U54-9 goes high at the end of the refresh cycle. A memory cycle begins when U54-8 goes low. The duration of the memory cycle is determined by one-shot U50-4.

All timing information given on the schematic is for reference purposes only to aid in following the operation of the schematic circuits. The memory should be timed according to the detailed timing specifications shown in Fig. 4. The value of an occasional one-shot timing resistor may have to be changed to obtain this timing. Timing not shown in Fig. 4 is not critical and should be obtained using the component values given on the schematic.

Seven address inputs are simultaneously applied to each row of memory chips. These addresses come from the address multiplexing circuits (see Fig. 5). Fourteen address inputs (A0 through A13) from the computer are separately applied to the memory chips in two steps, using the address multiplexing circuits, to select a memory chip address. In addition, seven refresh addresses are generated by the refresh counter, which advances by one count with each refresh cycle. The refresh address counter repeats after every 128 refresh cycles.

The memory address decoder (U44) decodes the memory addresses in 8K blocks throughout the address range (see Fig. 6). 64K is decoded using eight decoder outputs. Address bits are asserted on the bus when they are brought low. All address bits will be high for the lowest address count (000 000). All address bits will be low on the bus for the highest address count (377 377). 8K decoding blocks are necessary to disable the memory when addresses are selected in the lowest 8K range. The SEL-H line connects to U44-7 to accomplish this.

The decoder outputs are then paired using U43 to provide a 1-out-of-4 decoded address (four 16K decoding blocks) to select one of the four rows of 16K memory chips. The memory RAS timing signal, MCS-H, is applied to only one row of memory chips at a time. This is the address-selected memory chip row. Conversely, the refresh RAS timing signal, RRS-H, is applied to all memory chips simultaneously.

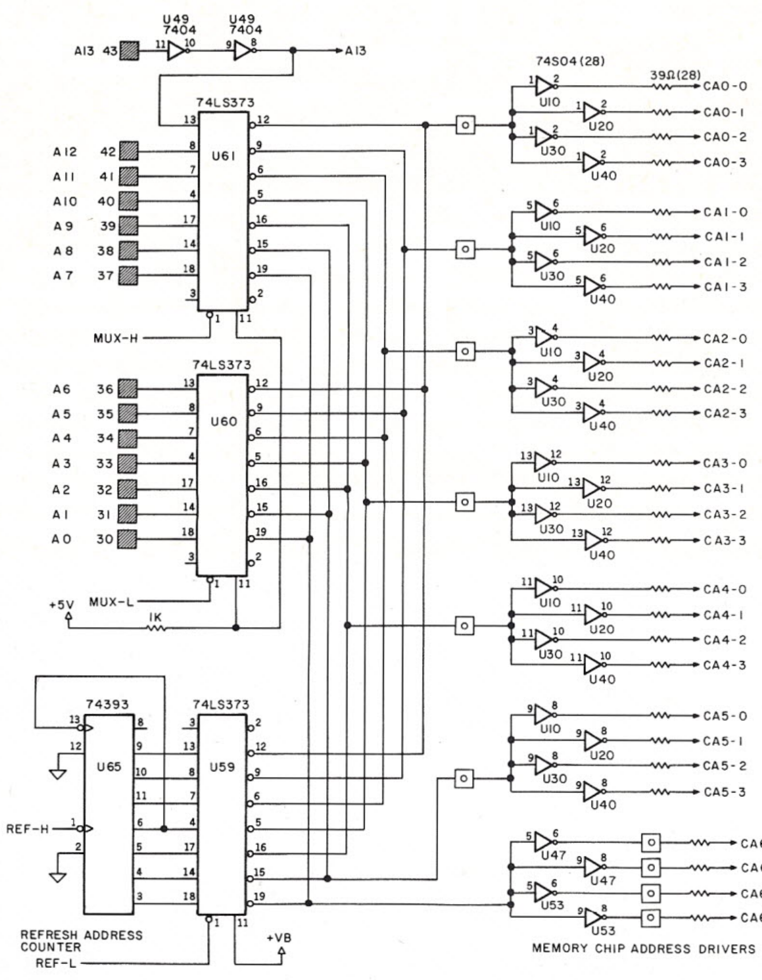


Fig. 5. Address multiplexing circuits.

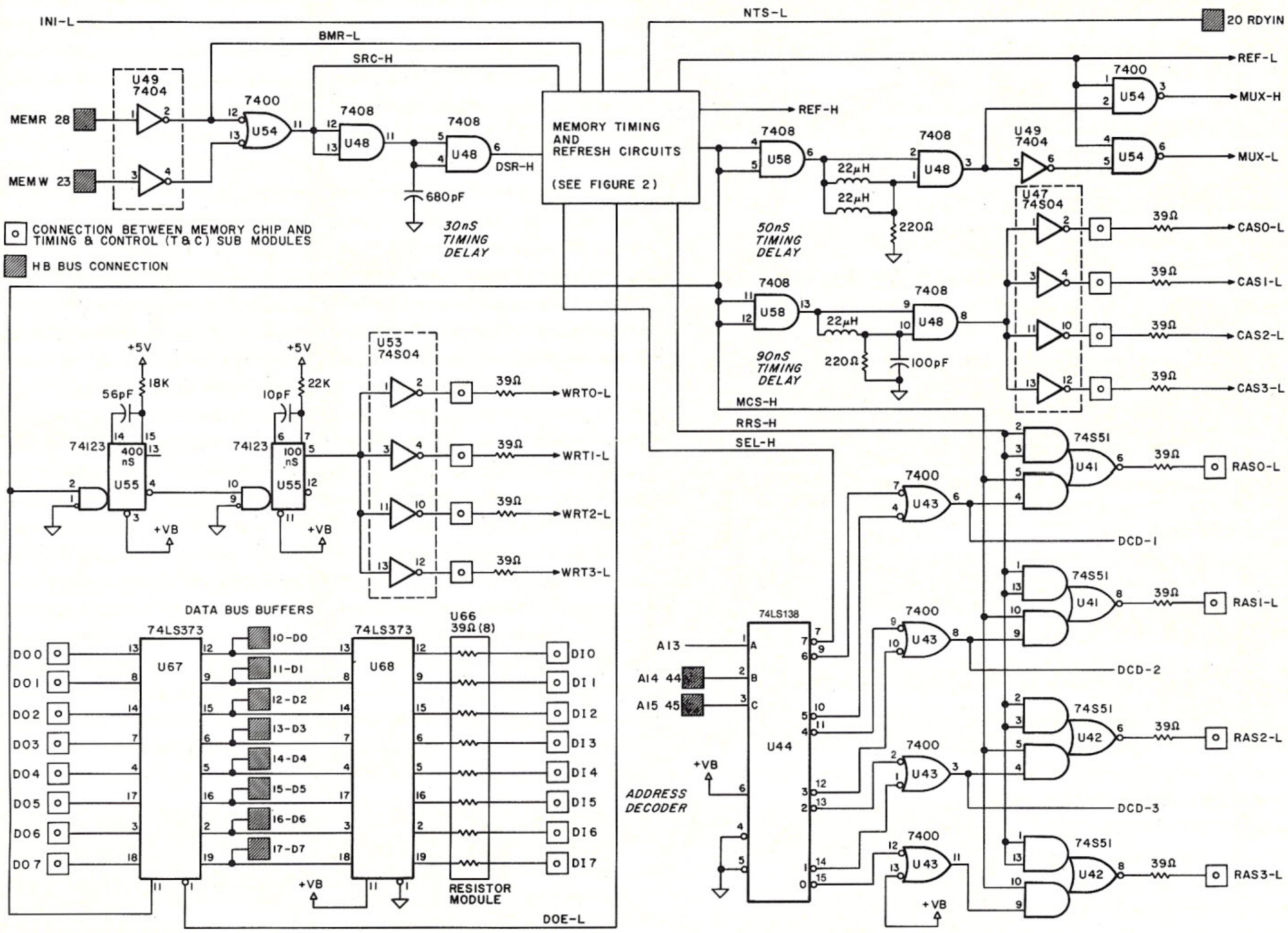


Fig. 6. Memory control circuits.

CAS and WRT are also applied to all memory chips simultaneously during computer memory cycles. Only RAS is applied to the memory chips during a memory refresh cycle.

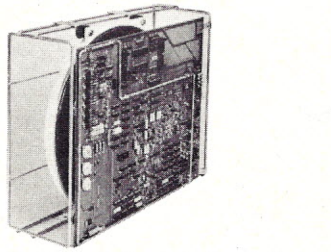
The memory chips are laid out in an array of four rows, with eight chips in each row (see Fig. 3). Respective address and control inputs are connected together in each row, which has separate address and control drivers in order to reduce the load on each driver. The dc input resistance to the memory chips is very high, whether they are driven high or low. However, the inputs have significant capacitance, which requires good drivers and series damping resistors.

The MOS memory chip inputs are TTL compatible; however, they are not the same as TTL inputs.

All the respective memory chip power supply voltages are connected together. This is not shown on the schematic, but it is understood. The memory chip data inputs and outputs are connected together by column, rather than by row. There are eight columns of data input and output lines, with four memory chips in each column. The data lines are thus connected to the memory chips in opposition to the address and control lines. In this manner, the memory chip array is formed.

This concludes the actual design of the memory for the Heathkit H8 computer. Next month, in part 2, we will consider the construction, operation and troubleshooting of the memory board. ■

PRIAM Hard Disks Now Available from SIRIUS SYSTEMS!



PRIAM's high-performance, low-cost Winchester disc drives speed up throughput and expand data storage from 20 megabytes to 154 megabytes. And a single controller can be used to operate 14-inch-disc drives with capacities of 33, 66, or 154 megabytes or floppy-disc-size drives holding 20 and 34 megabytes. So it's easy to move up in capacity, or reduce package size, without changing important system elements or performance.

- *Fast, Linear Voice Coil Positioning*
- *10 ms track-to-track positioning*
- *Fully servoed head positioning*
- *Dedicated servo tracks*
- *DC Power required only!*
- *Simple, parallel Interface*
- *Optional SMD Interface*
- *50 ms Average Positioning time*
- *90 ms Maximum Positioning Time*
- *6.4 ms Average Latency*

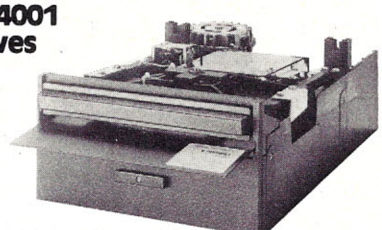
THE PRIAM LINEUP

Model/Disc Size	Capacity	Size	Weight	Price
DISKOS 3350 (14")	33Mbytes	7" × 17" × 20"	33 lbs.	**$2995**
DISKOS 6650 (14")	66 Mbytes	7" × 17" × 20"	33 lbs.	**$3749**
DISKOS 15450 (14")	154 Mbytes	7" × 17" × 20"	33 lbs.	**$4695**
DISKOS 2050 (8")	20 Mbytes	4.62" × 8.55" × 14.25"	20 lbs.	**$2995**
DISKOS 3450 (8")	34 Mbytes	4.62" × 8.55" × 14.25"	20 lbs.	**$3745**
DISKOS 570	5.3 Mbytes	floppy-size	(low)	**(low)**
DISKOS 1070	10.6 Mbytes	floppy-size	(low)	**(low)**

All PRIAM DISKOS Drives have a Transfer Rate of 1.03 Mbytes/Sec.
Optional SMD interface available for $150.

SIRIUS SYTEMS offer cases and enclosures for all PRIAM Hard Disk Drives. All 14" Winchester Drives will mount in our 14" Standard Case. The 8" Winchesters have two alternatives: a single drive case and a dual drive case. All SIRIUS SYSTEMS Winchester drive cases include Power Supply, internal cabling, switches, fan, extra AC outlet (not switched, but fused) and possess very adequate ventilation. Drive addressing is done on the rear of the Case and not on the drive iteself to provide ease of use during operation. All WINCHESTER DRIVE Cases are Warranted for a full year and come in our standard blue-black color scheme. Consult us for current availability and pricing.

Remex RFD 4000/4001 8" Floppy Disc Drives Double sided... Double density!!



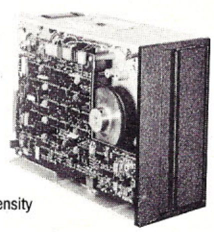
$549.95

RFD 4001, $569.95

Offers quality and features found in drives costing much more! ■ Single or Double Density ■ Double-Sided Drive ■ Door Lock INCLUDED ■ Write-Protect INCLUDED ■ 180 Day Warranty ■ Compatible with Shugart 850/851 ■ Low Power Operation ensures LONGER LIFE!! ■ Model RFD 4001 offers Data and Sector Separator

RFD 4000/4001 Technical Manual **6.95**
Connector Set #3 (AC, DC, Card Edge) .. **10.95**
Connector Set #4 (AC and DC) **2.95**
RFD 4000C/B Cabinet (for use with Power Modules) **29.95**

Remex 1000B... If you've been looking for a less expensive floppy disc drive, but not wanting to sacrifice quality — this is it!



$419.95

You get both in the Remex 1000B! For only $419.95 look at what you get: ■ 8" Floppy Drive ■ Single or Double Density ■ Hard or Soft Sectoring ■ Media Protection Feature ■ Single Density Data Separator ■ 180 Day Factory Warranty

Door Lock Option **$19.95**
Interface Adapter (REMEX-to-Shugart) ... **$14.95**
Write Protect Option **$19.95**
Connector Set #1 (AC, DC, & Card Edge) ... **$10.95**
RFD 1000B Technical Manual . **$5.95**
RFD 1000B CASE (for use with Power Modules) **$29.95**

SIRIUS 8" DISK POWER MODULES

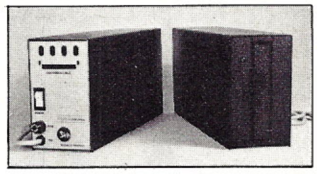
The Single and Dual Drive Power Modules are designed to provide DC and (switched) AC power for one (the Single Drive Power Module) or two (the Dual Drive Power Module—the DDPM will power three RFD 4000s or 4001s) 8" Floppy Disk Drives. Many features are included for safe and reliable operation and the Power Modules come with our standard 180 day WARRANTY (the Open Frame Power Supply warranty is for 2 years). All Power Modules will work with either the RFD 4000C/B or RFD 1000B case (color schemes match also).

Dual Drive Power Module (DDPM) **$139.95**

Single Drive Power Module (SDPM) **119.95**

SIRIUS 80+

Perfect Add-Ons for Your Computer System!



The SIRIUS SYSTEMS 80+ Series of Floppy Disk add-ons are designed to provide unmatched versatility and performance for your computer. Consisting of four different add-ons, there is a 80+ Series Floppy Disk to meet your need. All 80+ Series Floppy Disk are compatible with the TRS-80* and come ready to plug in!

COMMON CHARACTERISTICS

- *5 ms track-to-track access time*
- *Auto-eject*
- *180 day WARRANTY*
- *Exceptional speed stability — 1½%*
- *Single density (FM) or double density (MFM/M²FM)*
- *Ultra high reliability*
- *2 year Power Supply Warranty*
- *Mix any or all 80+ Series on the same cable!*
- *Includes user accessible plugboard for drive reconfiguring*

SPECIFIC CHARACTERISTICS

The **SIRIUS 80+1** is a single sided, 40 track, highly reliable Floppy Disk add-on. Offering 5 more tracks than the Radio Shack model, it cost $140 less! Formatted data storage is 102K/20K bytes single/double density.

SIRIUS 80+1 .... **$359.95**

The **SIRIUS 80+2** is a dual sided, 70 track (35 per side), highly versatile Floppy Disk unit. It appears to the TRS-80* as TWO 35 track drives, yet COST LESS THAN HALF THE PRICE! Even greater savings result, since data is recorded on both sides of the media instead of only a single side. Using the plug board, it may be reconfigured for other computer systems! (The 80+2 operates as Drive 0 and any of the other three addresses (with the sandard Radio Shack Cable) or as any of four drives (with the SS Standad Cable).) Formatted data storage is 80.6K/161.2K bytes single/double density.

SIRIUS 80+2 .... **$449.95**

The **SIRIUS 80+3** is a single sided, 80 track, "Quad" density Floppy Disk unit. Offering 2⅓ times the storage of a Standard Radio Shack drive, the 80+3 greatly reduces the need for diskettes correspondingly. Additionally, because of the increased storage and faster track-to-track access time, the 80+3 allows tremendously increased throughput for disk based programs!!! The 80+3 INCLUDES SIRIUS's TRAKS-PATCH on Diskette. Formatted data storage is 204K/40K8 bytes single/double density.

SIRIUS 80+3 .... **$489.95**

The **SIRIUS 80+4** Floppy Disk add-on is a double sided, 160 track (80 per side), 5¼" monster! The ultimate in state-of-the-art 5¼" Floppy Disk technology, to 80+4 is seen by the TRS-80* as two single sided disk drives, each with 80 tracks. Thus, in terms of capacity one 80+4 is equivalent to 4⅔ standard Radio Shack drives — a savings of over 73% (not to mention diskettes!!!). (With a double density converter, the available memory is huge!) The 80+4 is similar to the 80+2 in that it arrives configured as Drive 0 and any of the other three addresses (with the standard Radio Shack Cable) or as any of four drives (with the SS Standard Cable). The 80+4 INCLUDES TRAKS-PATCH on Diskette. (The plug board is also included.) Formatted data storage is 408K single density or 816K bytes double density.

SIRIUS 80+4 .... **$624.95**

0

All 80+ Series Floppy Disk add-ons operate a 5 milliseconds track-to-track access time (eight times faster than the SA 400) but are Expansion Interface Limited to 12 milli-seconds for the TRS-80*.

*TRS-80© Tandy Corp.

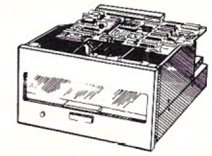
MPI 51/52...

A Great Reliable Mini-Drive!

- *Fast! 5ms track to track access*
- *Exclusive Pulley-Band Design*
- *Unique Door/Ejector Mechanism*
- *Reliable 1½% Speed Stability*
- *Single/Double Density Operation*
- *Industry/ANSI Standard Interface*

MPI 51
(Single Head, 40 tracks, 120K/240K bytes Single/Double Density**) **$259.95**

MPI 52
(Dual Head, 70 tracks, (35/side), 218.8K/437.5K Single/Double Density**) **$349.95**



MPI 91/92... NEW STATE-OF-THE-ART DISK DRIVE!

MPI 91
(Single Head, 80 tracks, 240K/480K Single/Double Density**) **$389.95**

MPI 92
(Single Head, 160 tracks (80/side), 480K/960K Single/Double Density**) **$499.95**

***Unformatted data storage*

Introducing the Versatile, Low-Cost OMEGA Series Controller

As new technological advances bring down the cost of fast, reliable mass data storage, the need for an inexpensive, versatile controller have become greater and greater. To meet this need, SIRIUS SYSTEMS' OMEGA Series Controller was designed.

The SIRIUS OMEGA Series Controller Module utilizes an **on-board microprocessor** to mediate data transfer to a wide variety of peripherals from an equally wide variety of host computer systems. Up to four Winchester Hard Disks (8" or 14"), four 5¼" Floppy Disks Drives and/or up to eight 8" Floppy Disk Drives may be **in use at one time.** Host systems interfacing is accomplished via a parallel or a serial interface. With the additon of a Personality module, the OMEGA Series Controller Module is directly **compatible with many popular computer systems** (among them the TRS-80*, Apple, Heath, and others). Provision is made for the addition of a streaming tape drive, also.

SPECIFIC HARDWARE FEATURES INCLUDE:

- *Control of up to twelve Floppy Disk Drives (eight 8" and/or four 5¼")*
  - *8" and/or 5¼" Disk Drive Utilization*
  - *Single (FM) or Double (MFM) density data storage*
  - *Hard or Soft sectored diskette usage*
  - *Utilization of "Quad" density (96 tpi) 8" or 5¼" Disk Drives*
- *Control of up to four WINCHESTER type PRIAM DISKOS Disk Drives*
  - *8" or 14" may intermix on the same cable*
  - *Accommodates 8" and/or 14" drives of 5.3Mbytes to 154Mbytes*
  - *Ultra-Fast data transfers*
- *Extremely flexible host-controller interfacing*

SPECIFIC SOFTWARE FEATURES INCLUDE:

- Dynamic format modifications via command words
- Extremely flexible format acceptance for unusual data storage formats
- Easily interfaces to standard operating systems (TRS-DOS*, CP/M***, etc)
- Operates in either get/put sector mode or data string mode
- Performance parameters may be changed by EPROM replacement or Dynaminic Reprogramming

Dedicated systems cards are also available on a limited basis for the STD-BUS and the S 100. These cards feature shared memory also (again, software selectable) in addition to the regular OMEGA Series Controller Module features. Consult SIRIUS SYTEMS for current price and availability for the entire line of OMEGA Series Memory Units and Controllers. **Dealer inquiries are invited.**

TO ORDER CALL (615) 693-6583

Phone Orders Accepted 9AM-7PM (ESDT)

We accept **MC, VISA, AE, COD** (requires Certified Check, Cashier's Check or Cash) and **Checks** (personal checks require 14 days to clear). **SHIPPING AND HANDLING:** $7.00 per Floppy Disk Drive or 80+ Module ■ 5% for other items (any excess will be refunded) ■ **Foreign Orders** add 10% for Shipping & Handling. Payment in U.S. currency ■ Tennessee residents add 6% Sales Tax ■ **VOLUME DISCOUNTS AVAILABLE**

SIRIUS SYSTEMS
7528 Oak Ridge Highway
Knoxville, Tennessee 37921 ✓67

A Use for Misprogrammed PROMs

Don't throw away those misprogrammed 8223s!

George Young
Box 11
Auberry, CA 93602

I was recently looking for a decoder with six inputs to use to provide Chip Enables (CEs) for random access memory (RAM). The 75154 met the necessary requirements, but the 74154 comes in a 24-pin package, and I did not have enough "real estate" available to add a 24-pin package to the board.

The 7488/8223 had the necessary six input pins—five address lines plus the CE pin. It has 32 outputs, but only eight are mutually independent. It would provide only half the number of outputs of the 74154, but I did not need 16 outputs; eight were adequate.

As a high school teacher of electronics, I had quite a few misprogrammed PROMs "in stock." Students make errors programming them, and a single error ruins an 8223. However, don't throw away misprogrammed PROMs.

What to Do

The PROM matrix is depicted in Fig. 1. All we need is a group of lows still remaining in one of the four sections of the PROM. These do not even have to be sequential as indicated by the slanting squares in Fig. 1. We do need at least one different low remaining on each of eight consecutive rows of the matrix.

With pins 12, 13 and 14 all low, we will access the eight rows at the top of the matrix. Pin 12 high, pins 13 and 14 low will allow us to use the second group of eight rows. With pin 14 high, we are in the bottom half of the matrix. With pin 13 low, we access the third group of eight rows, while with pin 13 high, we access the fourth group of eight rows.

Once the group of eight mutual lows are found in the matrix, the rest of the cells are all burned high, pull-ups are added, and the address lines are used with, or without, inverters to access the one-quarter of the chip that will be used as a decoder with eight outputs. The pull-ups can even be omitted, and the decoder will still function.

The chip enables are active lows. If you require active highs, the eight outputs can be inverted with 7404 sections. You could program an 8223 for the eight active high CEs, but this would *not* be using a misprogrammed 8223.

At first glance, the usual response is, "What a waste!" But remember, we are using misprogrammed PROMs here that normally find their way into the "round file."

Two PROMs will provide CEs for 16K of address space in 1K increments, requiring less real estate and less power than a single 74154 (but not less power than a 74LS154).

Now don't you wish that you had saved all those misprogrammed 8223s? ■

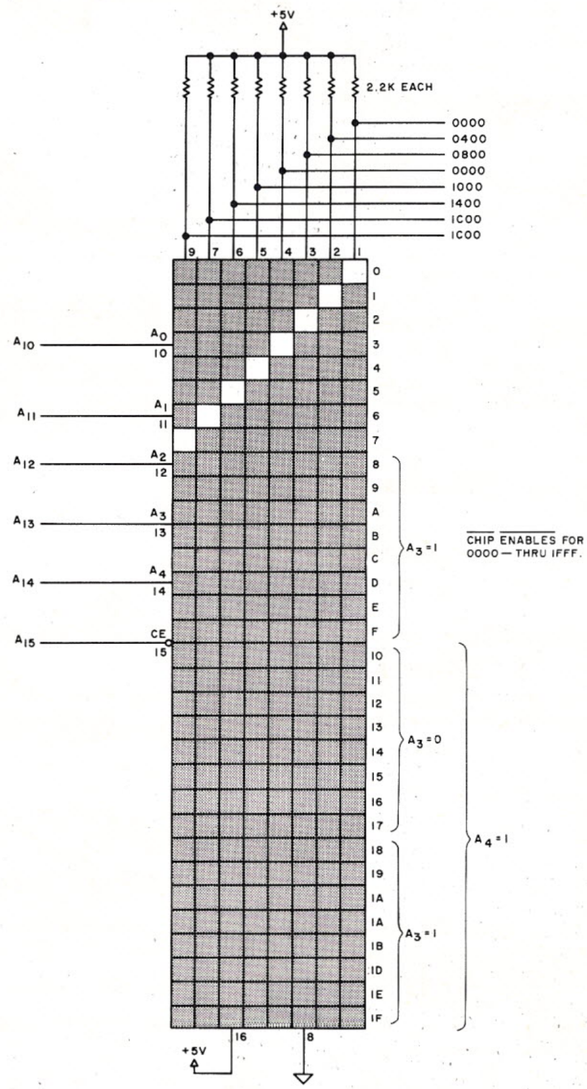


Fig. 1. 7488/8223 PROM matrix.

Thoughts on the SWTP Computer System

Like this series, 6809 systems are proliferating.

Peter A. Stark
PO Box 209
Mt. Kisco, NY 10549

The first SWTP 6809 system was the MP-09, which used the old 6800-style cabinet and a slightly modified MP-B2 motherboard. It is the latest 6800 motherboard, but is modified to place I/O at address E000 instead of 8000.

The S/09 system features a new cabinet, new motherboard, beefed-up power supply and even a fan. In this motherboard, all the cards—50 pin, as well as 30 pin—are parallel with the sides of the cabinet. The motherboard also has some new circuitry, including an interrupt timer, extended address decoding and a baud rate generator. The baud rate generator on the motherboard releases the baud rate lines on the 50-pin bus and lets them be used as address lines, so that the S/09 system comes with a standard 20-bit address bus.

Now the MP-09 system has been changed. It occupies the same cabinet as the S/09 but has an MP-B3 motherboard, which looks like the old motherboard, with the 50-pin cards parallel with the front of the cabinet and the 30-pin cards parallel with the sides. The larger cabinet allows the use of the new DMAF2 disk controller card, which was too large to fit into the older cabinet. It's also much sturdier than old SWTP cabinets, with 1/8 inch aluminum and hefty side rails.

But there is a big difference between these new MP-09 systems and the old MP-09s. The older system could use—with some changes—a 6800 CPU card and could run with the old SWTP mini-floppy controller and older I/O cards. On the other hand, the new motherboard will not work with a 6800 CPU card, nor will it work with the old mini-floppy disk controller (model DC-2). SWTP has now redesigned the controller, and the new DC-3 is just now becoming available.

> "The S/09 system features a new cabinet, new motherboard, beefed-up power supply and even a fan."

It is essentially a problem of address compatibility. SWTP 6800 systems assigned each I/O port four addresses and supplied two address lines to each port to allow the I/O card to select one of those four addresses. The new motherboard assigns 16 addresses to each port and supplies four address lines to each I/O board. This changes the board-addressing requirements so that some old boards will no longer work.

SWTP has, therefore, designed new I/O boards and eliminated the wiring tangle from the boards to I/O devices by placing the I/O connector on the rear edge of each card, so it is right up against the rear panel. They have also made other changes. They have replaced the MP-S serial interface with a $100 serial interface that has two ports instead of one. Also, EIA RS-232C has replaced the current-loop interface.

The new system has one other drawback: SWTP is no longer supplying schematics with their equipment, even to dealers. They prefer to have equipment shipped back to them for service. I wonder what the reaction to this will be in Europe, where the 6800 seems to be much more popular than the 8080.

6809 and 68000 Rumors

I teach computer technology

at a New York City junior college; our program concentrates on the 6800 because of its simplicity, as compared with some other processors. Among all our systems, we have eight SWTP systems and almost two dozen Heath 6800-based trainers. We plan on purchasing one 6809 CPU board to stay current but have tentatively decided to leapfrog over the 6809 and go directly to the 68000 in another year or so.

Richard Don of Gimix indicates that his company is already looking forward to the Motorola 68000 chip, which will use 16-bit data and include features rivaling today's minicomputer. The question is, "What systems will use it?"

Since there are so many SS-50 systems, many of the SS-50 suppliers are interested in offering a 68000 system that will be compatible with existing hardware. This is no minor matter, since all existing buses use 8-bit data, whereas the 68000 will use 16-bit data.

Alpha Micro, with its 16-bit processor and the S-100 bus, is multiplexing the sixteen data bits over the existing 8-bit data bus. This allows existing memory and other boards to be used with the new CPU, though it slightly slows down operation. A similar idea could be applied to the SS-50 bus and the 68000.

My suggestion is to design a new SS-50 motherboard with the 68000 CPU board occupying two slots in the center of the board, and eight of the data bits going to one end of the motherboard, while the other eight go to the other end. In this way, the 16-bit memory data could be stored in parallel in two 8-bit memory boards.

Whichever way it goes, though, one message is clear: Many manufacturers presently involved in making SS-50 bus equipment want to keep the 68000 compatible. And that should be welcome news to those of us with a heavy investment in 6800 hardware.

Memory Expansion

Since last month's report on expanding the SWTP 32K memory board, two friends have used the instructions to expand their boards from 16K to 32K, and everything has worked out well. In each case, they used a "16K memory add-on for the TRS-80" kit ($80) to add the extra 16K. You must make sure that you use fast chips (i.e., 250 nanosecond ICs) to make sure everything works well; otherwise, you may get errors.

Gimix also has a 32K board that is fully static. Using 2114 low-power chips, it takes only about 1.8 Amps, and it's reasonably priced at $548.15. It will work at 2 MHz; the SWTP 32K dynamic board will not. (Since the SWTP 16K/32K board does all refreshing during phase 1 of the clock, actual reads and writes are limited to using only the phase 2 portion of each cycle. At a CPU speed of 1 MHz, this leaves less than 500 nanoseconds for an entire read/write cycle; this is tight, but quite workable as long as the memory chips are fairly fast. At 2 MHz, the time available would be less than 250 ns, requiring memory chips with access time of perhaps 100 ns, or even less.) This is important to users of the Smoke Signal Broadcasting 8 inch floppy-disk system, which requires computer operation at the 2 MHz speed.

ROM Notes

If you want to use a 2716, but only need 1K of EPROM, consider the 2758. It's a "defective" 2716, with just 1K of usable memory. Sound familiar? It can be programmed with the SWTP 2716 programmer and used in the MP-A2 CPU board. Some suppliers sell it for a higher price than a real 2716. But that may change.

Pin 19 of a 2716 is normally the most significant address bit, A10. On the 2758 it is renamed AR and called a "voltage reference." For the standard Intel 2758, this pin should be held at ground, but there is an alternate version, 2758-S1865, which requires +5 volts on AR. When you think of it as an address pin, you realize that the standard 2758 has a good lower 1K, always requiring A10 at 0, while the 2758-S1865 has a good upper half that requires A10 at 1.

More Memory Expansion

Later on, I'll describe a need for some scratchpad RAM, out of the way of other programs. The ROM monitor, of course, already has a scratchpad RAM in the form of a 6810 on the CPU board, but often you need an even larger scratchpad. How can we get such an enlarged scratchpad memory at the least cost?

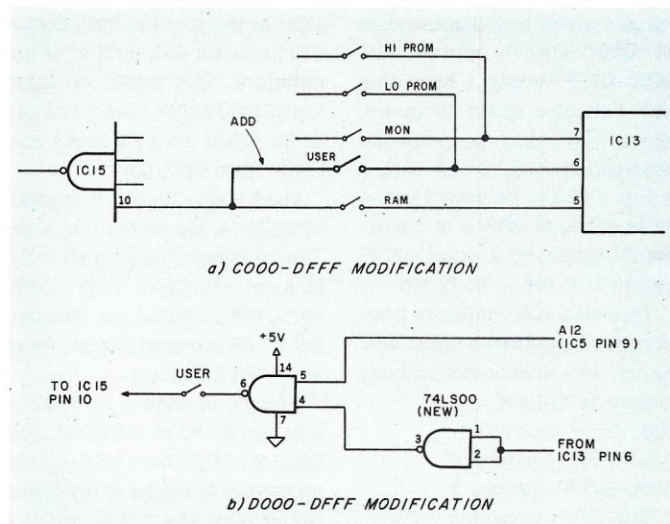


Fig. 1. Moving CPU scratchpad RAM.

Although there is a 6810 RAM memory on SWTP CPU boards, many systems have the 6810 disconnected to leave room for a 4K or 8K memory board located at $A000 for use with a disk operating system, such as Flex 2.0 or Percom's MINIDOS-PLUSX.

You can reconnect the 6810 on an MP-A2 CPU board with two wires in about five minutes, but readdress it elsewhere. You can use this 128-byte memory area—safely away from other memory where it might be clobbered by BASIC or other programs—for your own monitor or other routines.

The simplest way is to move it to lie at C000 (though this is only usable if addresses C000–DFFF are unused). It will then occupy C000–DFFF (just as it used to take up A000–BFFF). The modification uses the USER position of the DIP switch located on the CPU board. Just wire the USER switch as shown in Fig. 1a. With the RAM switch closed, the RAM will be at A000; with the USER switch closed, it will be at C000. Just remember never to close both switches simultaneously.

With this modification, this 128-byte scratchpad RAM will occupy the region from C000 through DFFF, reappearing in this 8K address space 64 times. That is, location C000 is the same as C080, C100, C180, C200 . . . all the way up to DF80. This wastes space because the RAM address is incompletely decoded; this is the same case with the normal CPU scratchpad RAM at A000–A07F.

The monitor program I'll discuss later uses additional RAM at D000. This memory at C000–DFFF is just the thing for that, since the decoding makes it also appear at D000. If address C000 is already occupied, then the modification changes (Fig. 1b). This time, the RAM is selected only if bit A12 is 1, which occurs for addresses starting with D, but not C.

It is also possible to expand the existing 128-byte RAM by substituting other memory ICs instead of the 6810. A small add-on board that allows two 2114 RAMs to be plugged into the 6810 socket to provide 1K of memory at A000 (C000 or D000 with the above modification) is available from Bruce Naples, PO Box 192, Oakland Gardens, NY 11364. A complete kit, including board, ICs and connectors, costs $30, postpaid.

You can also use a 4K board

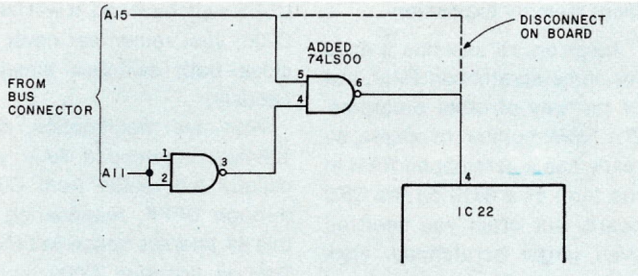


Fig. 2. 2K modifications to 4K board (74LS00 pin 14 to +5 V, pin 7 to ground).

as a 2K board. This may sound strange, but it's useful. I have a Percom video board addressed at D800-DFFF, which left D000-D7FF empty. I filled this with half of a spare 4K board, which was usable as an added scratchpad. The circuit shown in Fig. 2 has to be added to the 4K board to disable it in the upper 2K range, so it does not interfere with the video board.

This mod automatically positions the board in the upper 32K, so just wire the board's address jumper as follows:

9000-97FF—jumper 1
A000-A7FF—jumper 2
B000-B7FF—jumper 3
C000-C7FF—jumper 4
D000-D7FF—jumper 5

If you have a video board, avoid the temptation to use an unused part of its RAM as a scratchpad RAM. In general, whenever the CPU is accessing video board RAM, this same RAM is not available to the video circuits, and so the screen has a small blotch due to a missing character. Normally, this happens only when new video data is being fed to the board, but if you use part of the RAM as a general-purpose scratchpad RAM, then you may get a lot of interference with the display.

Percom LFD-400 Modifications

A few months ago (February 1980), I suggested that you use your 2708 programmer to modify Percom's LFD-400 disk operating system EPROMs. I have since then modified mine even more.

One modification increases the size of the disk directory from two to four sectors, which provides for up to 63 files, rather than the standard 31 files. This requires changing the MINIDOS-PLUSX EPROM (A2 to A4 in locations C4FC, C520, C567, C69A, C6BE, C702, C77C and C78B).

At the same time, I moved the start of the first file from sector 010 to sector 004, right after the directory. This again changes MINIDOS-PLUSX (C6C3 from 01 to 08, C6C6 from B9 to BA and C6DF from 06 to 00).

Next I substituted > instead of @ for a file protection code. This required changing 40 to 3E in locations C563, C59F, C735 and C769. I needed this because my CT-64 terminal doesn't have @ on the keyboard.

Finally, changing 22 to 27 in location C17C of MINIDOS and C608 of MINIDOS-PLUSX opened up the full 40 tracks of my drives to the DOS. The last sector of a disk is now 399 rather than 349. Most Percom software, except the Percom Assembler, which doesn't go past sector 349, relies on the DOS for sector checking and so works well with 40 tracks.

These changes gave me 396 usable disk sectors instead of 340, an increase of 16½ percent from 87,040 bytes to 101,376 bytes.

When I first received my Percom LFD-400, I was using a 2716 EPROM in the C000-DFFF area, so it was inconvenient to use C000 and up for the disk. I therefore made a slight modification on the disk controller board, which moved the whole board from C000 to 9000. This involved cutting the traces that bring A14 and A12 to B10 and B11 on the controller and interchanging the address bits so that the address decoding changed from

A15	A14	A13	A12	
1	1	0	0	=C

to

A15	A14	A13	A12	
1	0	0	1	=9

This moved the 2708 EPROM on board the disk controller to 9000 and up and moved the actual disk control addresses from CC00 down to 9C00.

At this point, I was faced with modifying all Percom software that used DOS routines to the new addresses, which seemed a formidable task. Instead, I burned both MINIDOS and MINIDOS-PLUSX into a single 2716 and left it at C000, but on the CPU board. (Needless to say, this required changing all disk addresses within the monitor from the CC range to 9C.) Everything was now compatible with Percom software, none of which accesses the disk controller directly, but left another 6K open above C800 for more EPROM.

I have recently returned back to the original addresses (since I now have a 2708 EPROM board at E000 and RAM at 9000), but this same modification may be still useful to others.

Percom has released a software item that allows the LFD-400 drive to run either mini-Flex or Flex 2.0. This program reads any Flex soft-sectored diskette on one Percom drive and copies it onto a ten-sector hard-sectored diskette on the other drive; then it substitutes different disk drivers into Flex. Flex then runs exactly the same as if it was on an SWTP disk system, with the only difference being that it uses hard-sector diskettes instead of soft-sector diskettes. Thus, your disks won't be usable on someone else's SWTP disk system, although you can translate regular Flex disks into the Percom format and use them.

Having used the system, I'm impressed. It runs faster than Flex on an SWTP disk and is otherwise exactly the same. In fact, some of the Percom software can still be used—you have a choice of using either the Flex Backup or the Percom Backup routine to copy Flex disks. Percom's is much faster!

There is also a version for running Smoke Signal Broadcasting's DOS on a Percom system.

Another Percom product is an improved data separator that plugs into the SWTP disk controller to improve the read error rate. Though designed mainly for the TRS-80, it also speeds up the SWTP disk system. The reason is that when the verify option is used, each write to a disk is followed by a read to check that writing was done correctly. This slows down disk operation, especially when read errors occur, since the DOS will reread the sector several times when an error is detected. Reducing read errors by adding this Percom data separator should, therefore, speed up not only reading, but also writing. I was told that the difference is very obvious.

Tape Read Errors

Occasionally, someone sends me a Kansas City tape that I absolutely can't read. This is usually due to distortion or bad head alignment in making the tape. I have recently found a solution which helps most of the time.

In the August 1978 issue of *Kilobaud Microcomputing* ("Copying Computer Cassettes," p. 94), I described a device for copying Kansas City tapes. This data regenerator takes a tape signal and cleans it up so it can be rerecorded. Inserting this circuit between my cassette player and the input to my cassette interface has helped me read many tapes that I could not otherwise decipher (printed-circuit boards for the circuit are no longer available).

Monitoring Cassette Loading

When you load a machine-language cassette program using the monitor's L command, the monitor turns off the echo function so that you cannot see what you are reading on your terminal. At the same time, it turns on the reader control line to switch the cassette interface from keyboard to tape input.

If you don't need the reader control signal, then it is easy to enable the echo. In SWTBUG, instead of typing in the L command, simply jump to LOAD3 in the monitor by typing J E00F. This will bypass the part that turns off echo. Another way to do this automatically is to put a segment that loads a 00 into location A00C (SWTBUG's PORECH location) at the beginning of the tape being loaded.

dventures are
isruptingly addictive
irtually thrusting
gos into
arcosis — beware!
ime itself
nravels while
isking these
nchantments by
cott Adams.

Great Scott! What have you created?

The international reception of Scott Adams' ADVENTURE game series is phenomenal! Each of the ten games takes the player on odysseys of intrigue and danger, through confrontations with various villains, in pursuit of a wide variety of mysteries and treasures, in surroundings both familiar and alien. You may find yourself in a voodoo den, on the edge of a strange and threatening galaxy, in the core of a malfunctioning nuclear reactor, or in any one of the several other settings; ranging from mundane to bizarre.

Challenge your genius! Order an ADVENTURE

Master these ADVENTURES!

0. Special Sampler
1. Adventureland
2. Pirates Adventure
3. Mission Impossible
4. Voodoo Castle
5. The Count
6. Strange Odyssey
7. Mystery Fun House
8. Pyramid of Doom
9. Ghost Town

ADVENTURES are available for Pet™, Sorcerer™, Apple™, TRS-80™ Microcomputer, and other major personal computers.

✓287 **adventure international**
Box 3435 • Longwood, Fl 32750
(305) 862-6917

Look who's raving about ADVENTURE

Out of 50 programs reviewed, Adventure was rated No. 1! "Highly recommended."
80 Software Critique

"Adams' Adventure is exquisite. It is a true tour-de-force . . . "
Recreational Computing

" . . . It is worth the money to buy a computer just to play Adventure."
Software Directory Winter 1980

" . . . Truly absorbing simulations. These games require you to be inquisitive, innovative, a thinker, a risk taker, a logician, a warrior, and a lover of real challenges — in short, an Adventurer."
Ramon Zamora, Recreational Computing Issue 4

"I LOVED IT! There was a real challenge to this program . . . "
Captain 80, 80 Microcomputing Issue 1

Here's a few of the more than 200 ADVENTURE dealers.

Level IV Products, Inc.
32238 Schoolcraft, Suite F-4
Livonia, MI 48154
800-521-3305

Computerland Salt Lake
161 E. 2nd South
Salt Lake City, UT 84111
801-364-4416

Computer City
175 Main St.
Charlestown, MA 02129
800-343-6522

Program Store
P.O. Box 9609
Washington, D.C. 20016
202-337-4691

Computerland Downers Grove
136 Ogden Ave.
Downers Grove, IL 60515
312-964-7762

Computronics
50 N. Pascack Rd.
Spring Valley, NY 10977
914-425-1535

✓ Reader Service index—page 241

Unfortunately, there is no easy way to do this when using the O L command.

CPU Clock Speed

The CPU clock in the older MP-A CPU board is crystal controlled at just under 900 kHz. The clock frequency in the newer MP-A2 board is, however, controlled by an RC network, which has a tendency to drift all over the place. Most of the time it doesn't make much difference, but there are some cases (such as when using the JPC 4800 baud cassette interface or some disk systems) where clock speed is important.

It is easy to make the MP-A2 crystal-controlled. Simply remove C1, the timing capacitor connected to the 6875 clock IC on the CPU board, and substitute a 3.579 MHz color TV crystal instead. Because of the division by 4 in the 6875, the resulting clock speed will be one quarter of that, or just under 900 kHz. Although you could use a 4 MHz crystal instead, 3.579 MHz crystals are inexpensive and easy to obtain (I got mine for 99 cents at a local parts emporium). You could also use 80-meter ham crystals around 3.7 MHz, as long as they are the newer, metal-case types, not the old, pressure-mounted ones.

You can easily use a frequency counter to measure the actual clock frequency, but an easier way is to use a calibrated program. Listing 1 is written to run exactly 100 seconds—from the time you start it until it rings the bell on your terminal, if the clock speed is exactly 1 MHz. Simply run the program, time how long it takes and compute your clock frequency from the following formula:

$$\text{clock freq. (MHz)} = \frac{100}{\text{time (sec)}}$$

For example, with a 3.579 MHz crystal controlling the clock rate, running time for this program is 112 seconds, which results in a computed clock rate of 893 kHz. The actual clock frequency is 895 kHz (3579 divided by 4).

```
                         NAM    TIMER
     (0100)              ORG    $0100
0100 CE 0000    START    LDX    #$0000
0103 86 BF               LDA A  #$BF
0105 09         LOOP     DEX
0106 26 FD               BNE    LOOP
0108 4A                  DEC A
0109 26 FA               BNE    LOOP
010B 86 07      DONE     LDA A  #$07
010D BD E1D1             JSR    $E1D1
0110 7E E0E3             JMP    $E0E3
                         END
        00  ERROR(S) DETECTED
```

Listing 1. CPU clock speed timer.

6800 Monitors

Table 1 compares eleven ROM monitors, with which I am familiar (data for this table was adapted, with permission, from a comparison table published by Gimix):

GMXBUG versions 3.0, 2.0 and 1.0, from Gimix, Inc., 1337 West 37th Place, Chicago, IL 60609. All three versions are designed to be used with the Gimix 6800 board and the Gimix video board. Another GMXBUG is version 3.0/8KP, a special version for non-Gimix CPU boards.

SWTBUG is by Southwest Technical Products Corp., 219 W. Rhapsody, San Antonio, TX 78216.

RT-68 is by Microware Systems Corp., PO Box 4865, Des Moines, IA 50304, and is unique among all of these since it offers real-time capability to switch computer time between a number of tasks, which take turns executing.

MIKBUG is *the* original Motorola monitor.

SMARTBUG is available from

FEATURE	GMX BUG V3.	GMX BUG V2.	GMX BUG V1.	SWT BUG	RT/ 68	MIK BUG	SMA-RT BUG	DOO-DLE BUG	MSI BUG	JOE BUG	SMI-TH BUG	HUM BUG
Monitor size	3K+	3K	2K	1K	1K	512	1K	1K	1K	2K	2K	3K
ROM Type	´08	´08	´08	´30	´30	´30	816	´16	´08		816	816
Add to SWTBUG								no			yes	
Replace SWTBUG	yes	yes	yes		yes		yes	no	yes	yes	yes	yes
Control port	4L			1SC	1C	1C	2S	2S	0S	1S	1S	1S
Alt. control port					2S					7L		
Other I/O ports	3			0SC				7L	1S	1	1	2
Video board	GMX	GMX	GMX							THO		PER
Video format	24x	16x	16x							16x		16x
(lines*char)	80	64	64							64		80
MEMORY EXAM entry	M	M	M	M	M	M	M	M	M	M	E	ME
Exam next loc	+	+	+	sev	LF	sev	SP		sev	sev	SP	sev
Exam previous	-	-	-	↑			U		.	↑	↑	↑
Change & advance	#	#	#	#	/	#	#	#	#	#	#	#
Change, no adv	=#	=#	=#									
Display binary	2	2	2									
ASCII entry	"$	"$						S			at	AI
OTHER MEMORY FUNCTIONS												
Hex Dump	D	D	D		D			D			V	HD
Dump Assy Lang											D	
Dump Mach Lang	F							E	T			DE
ASCII Block out	D	D	D									
ASCII Text out												AO
Move Memory	X	X								C	M	MO
Fill Memory	Z	Z					I			F	I	FM
Find byte in mem	H			F				1-3		A	F	FI
(#of bytes)	1-3			1				1-3		2	1	1-3
Memory test	T	T	T							V		MT
Init Stack Ptr	I	I	I									
Init IRQ vctr								I				
Init NMI vctr								N				
DEBUGGING FUNCTIONS												
Set Breakpoint	B	B	B	B	B		K			B	1,2	BR
(# of Bks)	4	4	4	1	1		1			1	2	4
Reset Breakpoint	K	K	K	B	B							BR
Print Brk Locs.	P	P	P									BP
Print Registers	R	R	R	R	R	R	R	R	R	R	R	RE
Change Registers							yes	cc			yes	
Checksum memory	C	C	C						C			CS
Single-Step							T				T	SS
Hex Arithmetic	A	A	A									
EXECUTE USER PROGRAM												
Goto User Pgm	G	G	G	G	G	G	G	G	G	G	G	GO
Jump to User Pgm				J	E		J			J	J	
JSR to User Pgm	J	J	J									JU
Cont. Frm Break				G	G						K	CO
Jump to C000 ROM	U	U	U	Z	Esc					Z		PD
Jump to E400 ROM							4					
Jump to E800 ROM								8				
TAPE SAVE/LOAD												
Load from tape	L	L	L	L	L	L	L		L	L		LO
Load w. offset	L	L	L									
Punch Tape	S	S	S	P	P	P	P		P	P		PU
Punch PC and S9				E						E		EN
Binary Pun/Ld								P,L		T,S		
Tape on Port 0				0					yes			
DISK BOOTSTRAP												
SWTP 5"		Q		D						D		FD
Flex Warmst.										H,W		
SSB 5"							Q				Q	
SSB Warmst.							D				&	
SSB 8"										U		
Percom Disk	U	U	U	Z	Esc		4			Z		PD
I/O PORT CONTROL												
Printer Subr.	I	O						yes		O	H	↑SO
Call Ext. Subr.							H					↑SD
Port 1 On/Off									↑E	Q		↑S1
Output Wait	↑S	↑S	↑S						↑S	↑S		↑S
CRT Pause												↑SP
Full/Half Duplex	I	H					E/N			N	O/N	
Clear Screen	E	E	E	C								CL
UTILITY ROUTINES												
MIKBUG Routines	24	24	4	22	17	All	37	0	14	32	27	16
Comp. Entry Pts.	24			22	17	All	37	0	14	32	0	16
Extended Rtnes	Yes	Yes	Yes		Yes		Yes					
R/T Multi Task					Yes							
OTHER												
User Def. SWI	Yes	Yes	Yes	Yes	No	No	No		Yes	Yes	Yes	Yes
Dumb Terminal										X		
Error Detection					Yes							
MEMORY USED												
ROM - Program	E0			E0	E0	E0	E0	E4	E0	E0	F8	E0
ROM - Vectors	Yes			Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
RAM Scratchpad	A0			A0	A0	A0	A0	A0	A0	A0		A0
and ...	6F!				00							D0
Alternate RAM										C0		

Table 1. 6800 ROM monitors.

Smoke Signal Broadcasting, 31336 Via Colinas, Westlake Village, CA 91361.

DOODLEBUG comes from Computerware Software Services, 656 Lomas De Oro, Olivenhain, CA 92024.

MSIBUG is by Midwest Scientific Instruments, 220 W. Cedar, Olathe, KS 66061.

JOEBURG is written by Joe Pentecost, Marietta, GA, and is distributed by Thomas Instrumentation, 168 8th Street, Avalon, NJ 08202. It is specifically aimed at supporting the Thomas Instrumentation Video Board.

SMITHBUG comes from Ed Smith's Software Works, PO Box 339, Redondo Beach CA 90277.

HUMBUG is my "Monitor to End All Monitors" effort. I will describe many of its routines, with source listings, in this series. A complete listing, however, takes almost 40 pages and can't be reproduced here. Hence, the full source code on disk or cassette, as well as 2708 and 2716 EPROMs, is available from Star-Kits, PO Box 209, Mt. Kisco, NY 10549. This monitor supports the Percom video board.

Monitor Features

Let's examine some of the features of Table 1.

Monitor size. The original MIKBUG occupied half of a 1K mask-programmed ROM and was, therefore, 1/2K in size. All of the newer monitors take up 1K, 2K or 3K. The more memory, the more functions.

ROM type. MIKBUG and SWTBUG both came on 6830 mask-programmed ROMs and plug directly into the CPU card. They are identified with '30 in the table, as is RT/68. Most others require EPROM, usually 2708 (labeled '08). Several (listed as 816) are available as either 2708 and 2716.

Add to/Replace SWTBUG. SMITHBUG can be used either by itself or in addition to SWTBUG (or MIKBUG). DOODLEBUG is not self-contained but must be used along with SMARTBUG. It is available either on SSB disk for use in RAM or along with SMARTBUG in the same 2716. All others are designed to completely replace the usual SWTP monitor.

Ports. This category represents a big difference among the various monitors. The table lists ports on four lines. The control port is the port used for normal monitor communication; an alternate control port is another port that can be used instead of, or in addition to, the main port. Some of the monitors also support additional ports that can be used for program output, and some also support a video board.

The original MIKBUG required a PIA port—in the form of an MP-C card—in port 1. This is shown in the table as 1C (1 stands for port 1, C stands for MP-C, S for MP-S and L for MP-L). This was both a control port as well as the normal program I/O port.

1SC in the SWTBUG column indicates that the main port in port 1 could be either an MP-C or MP-S card. For monitor P, L and E commands, SWTBUG also supports a serial device on port 0.

RT/68 requires an MP-C board in port 1 and can also use an MP-S card in port 2 in addition. SMARTBUG and DOODLEBUG require an MP-S card in port 2. DOODLEBUG also has provisions for a parallel printer on port 7.

GMXBUG is designed for use with the GMX video board for output and requires a parallel keyboard in port 4. It also will support three other I/O ports for program use—an MP-S for tape on port 0, an MP-L for a parallel printer in port 4 or an MP-S for a serial printer in port 3.

JOEBUG works with an MP-S card in port 1 as a control port, but when used with the Thomas Instrumentation Video Board, it will also work with a parallel keyboard on port 7 for input and the video board for output. It can operate with another serial or parallel port for program output.

SMITHBUG uses an MP-S card in port 1 for control but can call a user-written output handler for another output device such as a printer.

HUMBUG works with an MP-S card on port 1 and also supports an MP-S card in port 0, the Percom video board and a user-defined port for output.

The table also lists the number of lines and the number of

characters per line for those monitors that support a video board.

Monitors that do not use a serial input on port 1 introduce one major problem: a lot of software tests the keyboard on port 1 for a break character without going through the monitor. When a different control port is used or a parallel keyboard is used, then all this software has to be patched to work. For instance, the GMXBUG manual contains about 15 pages of patches to various programs so they will run with this monitor. This could lead to a serious incompatibility problem.

This is eliminated with an MP-S card on port 1. Next month I will show users of a video board another way—a special input card that interfaces a parallel keyboard on port 1 in such a way that all software thinks it's a serial keyboard connected via the standard MP-S—to solve this dilemma.

Memory exam functions. All of the monitors examine and change memory contents. In the remainder of Table 1, capital letters identify the monitor command for a particular function. In most of the monitors, memory can be examined or changed with the M command; in SMITHBUG it is done with the E command; and in HUMBUG it is done with the ME command (HUMBUG uses two-letter commands; all the others use one letter).

Let's take a look at SWTBUG's entries. M starts the memory exam mode. Several characters allow you to go to the next location. The up arrow (↑) allows you to examine the previous location. Entry of a number (shown as # in the table) allows a change. GMXBUG has a few more modes, such as change without advancing, display in binary or enter ASCII characters. SMITHBUG and HUMBUG allow ASCII character entry.

Other memory functions. Several monitors allow a hexadecimal, formatted dump of memory contents. SMITHBUG disassembles memory and outputs assembly-language mnemonics; GMXBUG, DOODLEBUG and HUMBUG output machine language, with address and one, two or three bytes of instruction per line. GMXBUG also allows a formatted ASCII output, while HUMBUG will dump ASCII text from memory.

The larger monitors provide additional memory functions, including moving a block of memory contents, filling memory with an arbitrary number (such as SWI), finding specified bytes in memory (one, two or three bytes), running a memory test and initializing the stack pointer.

Debugging functions. These are extremely important to assembly-language programmers but may be completely unused by BASIC programmers. Except for MIKBUG and MSIBUG, all other monitors allow the setting of a breakpoint. SMITHBUG allows two breakpoints, while GMXBUG and HUMBUG allow up to four. To keep track of where they are, the latter two monitors can also print out locations of current breakpoints.

When a breakpoint is encountered in a program, all monitors print out the contents of 6800 registers. This printout can also be requested with the R command. SMARTBUG and SMITHBUG also have specific commands for changing these registers, while DOODLEBUG allows changing the CC register. Additional debugging commands allow computing a checksum of memory contents, single-stepping through programs and performing hex arithmetic.

Execute user program. The two absolute minimum functions of a monitor are changing memory and jumping to a program. Hence, all monitors have a G or GO command to go to a user program. In most of these, address A048/49 is used as the pointer to the program. Several monitors also have a J or JU command to jump to a program without setting A048/49 first. In GMXBUG and HUMBUG, this command actually does a JSR to a user program, so that subroutines can be executed and will return to the monitor when done.

In some monitors, the G command is also used after a breakpoint to continue the program. In others, a different command,

such a K or CO, is used.

For use with external ROM programs, several monitors have a command such as SWTBUG's Z to jump to PROM. Most jump to address C000, though SMARTBUG jumps to E400, and DOODLEBUG jumps to E800.

Tape save/load. All monitors except SMITHBUG provide commands to punch and load tape using the standard MIKBUG format (S1 . . .). SMITHBUG doesn't have this function because it assumes that it is used as an adjunct to another monitor.

SWTBUG and several others have an E or EN command to punch the program starting address (A048/49) and an S9 at the end. SWTBUG also permits a port 0 to be used as an option, while JOEBUG and DOODLEBUG have binary punch and load routines for speedier operation.

Disk bootstrap. Almost all of the monitors allow disk bootstrapping in one way or another, but for different disks. If you already have a disk, this is important to you. JOEBUG is the most complete in this regard, as it has facilities for both the SWTP mini-floppy disk system as well as the SSB 8 inch floppy and Percom's LFD-400. (Since Percom's LFD-400 DOS is in EPROM at C000, the same command that jumps to user PROM at C000 is used for booting the Percom disk.)

I/O port control. This section lists the commands for controlling additional ports, if any. (↑S means control-S.) There are commands for enabling and disabling printer output, as well as for calling a user-written external subroutine. JOEBUG and HUMBUG have a command for turning off port 1 output while letting the video board continue at full speed. In several monitors, ↑S provides a wait function for stopping output while you read the CRT screen. HUMBUG also has a command for pausing automatically every 15 lines. The full/half duplex command controls echoing of input back to the output device.

Utility routines. In addition to monitor commands, most monitors have a set of subroutines, such as INEEE, OUTEEE and PDATA, as well as a host of less common ones, that are generally called by user programs for input and output. The original MIKBUG had several dozen such routines (and associated variables), which are often used by other software. For obvious reasons, other monitors provide the same routines, usually at the same addresses.

The utility routines section of Table 1 shows how many such MIKBUG routines each monitor has, and also whether they have compatible entry points. With the exception of GMXBUG version 1.0 and DOODLEBUG, all other monitors have most of the MIKBUG routines. Only about ten of the MIKBUG routines are commonly used, so most of the monitors will provide this basic core of routines. Likewise, most of the monitors provide these routines at the same entry addresses. The one exception is SMITHBUG, which has no compatible entry points. Keep in mind that SMITHBUG is intended to be used primarily as an addition to another monitor, not as a replacement for it. Hence, it relies on the routines in the other monitor. If SMITHBUG were used all by itself, major and extensive surgery would be required on other software to change monitor routine calls.

DOODLEBUG has no compatible routines, since it must be used with SMARTBUG and is not self-contained. Some of the monitors, such as GMXBUG, have extended routines not included in MIKBUG. GMXBUG is particularly rich in these, since it has routines to push and pull index register contents and add accumulators to the index register. Any program written to take advantage of these will not run with other monitors. These routines use the SWI instruction and cannot be used with Flex 2.0.

RT/68 is different from all the others in one big way: it supports real-time multitasking. An external clock signal, usually at a fixed rate such as 60 or 100 pulses per second, is connected to the control port to generate interrupts at each pulse; this is called a real-time clock. Each time an interrupt occurs, the program currently running is in-

terrupted, and control returns back to the "multitasking executive" portion of RT/68.

For each program (task) in memory at the time, there is an entry in an area of the scratchpad RAM called the "task status table," which contains various operating parameters for up to 16 programs.

Each time that an interrupt causes a jump back to RT/68, the monitor looks at the task status table to see whether the current task should be continued; if not, it determines which other task should be started. Thus, up to 16 tasks can be run. Some of these tasks might be independent programs, while others might be related to other tasks to work on the same problem.

Other. Some other features of the various monitors may be important in some applications. Most monitors allow the SWI instruction to be user-defined; only MIKBUG, RT/68 and SMARTBUG do not allow this. JOEBUG has an operating mode that allows the system to be used as a dumb terminal on-line to another computer system. RT/68 also has error detection; it will print out an error message if a command is used incorrectly.

Memory used. Most monitors are written to reside at address E000 and up; SMITHBUG starts at F800 because it may be used with another monitor and thus must use different addresses. DOODLEBUG's RAM versions are at either 1C00 or 3C00, and the 2716 version is at E400, which is the next 1K after the SMARTBUG with which it is used. The reset and interrupt vectors are required by the 6800 at addresses FFF8–FFFF.

Most monitors require just the standard 128-byte scratchpad from A000 to A07F; but the more versatile ones require more memory. Thus, GMXBUG uses memory from 6FFF down for a stack; RT/68 requires 12 bytes starting at 0000 only when multitasking (these are not needed in single-task use). HUMBUG requires 128 bytes at D000 in addition to the standard scratchpad at A000.

Finally, JOEBUG is available in an alternate version that has RAM at C000 instead of A000; this version is designed for use with the Thomas Instrumentation CPU board.

HUMBUG

Ever since I put a serial printer on port 0, I have been thinking of modifying my monitor to allow better control over it and also over the Selectric printer I have on port 4. When I recently got a Percom video board, the need to interface it properly to existing software became even more acute. The HUMBUG monitor was the result.

HUMBUG is an extended monitor intended as a SWTBUG/MIKBUG replacement. My standard version occupies three 2708 EPROMs and requires a separate EPROM board; a 2K version of HUMBUG, which fits into a 2716 EPROM and fits on the MP-A2 CPU board, also exists. This latter version does not have the Percom video driver and also omits the AI, MO and MT commands (see Table 1).

In designing HUMBUG, I had the following criteria in mind:

1. SWTBUG subroutine entry points had to be preserved so other programs would run unchanged. SWTBUG scratchpad addresses also had to stay the same.

2. I wanted full control over the system from the keyboard, including turning ports on and off in the midst of a program and aborting programs from the keyboard without pushing RESET.

3. I needed extended debugging facilities for machine- and assembly-language programming. I was not happy with the way SWTBUG handled breakpoints and stack pointers in user programs, since it would make an occasional annoying error. I also wanted a single-step mode for stepping through programs under test.

4. Finally, I expected HUMBUG to change with time and wanted to make it easily extended to new functions without major rework.

We'll discuss each of these criteria in turn next month. ■

slaving too long over a hot computer?

AUTOMATED SIMULATIONS

then it's time for a fun break!

The Temple of Apshai

First in the Dunjonquest™ series. Undertake heroic acts within a labyrinth filled with treasures and fantastic monsters who guard these treasures and move in real time. The Book of Lore (included) fills in the background and describes the appearance of the temple. Over 200 rooms and 30 monsters. There are 16 million kinds of characters! The best of the dungeon computer games.

For **TRS80** *Cassette*: 16K, Level II; *Disk*: 32K, TRSDOS • **Apple** *Cassette*: 48K Applesoft in cassette or in ROM; *Disk*: 48K Applesoft in ROM • **Pet** *Cassette*: 32K, old or new ROMs.

$24.95 cassette • $29.95 disk

Starfleet Orion

Fight space battles in your living room with 12 game scenarios (data files) using 2 to 15 spacecraft. Infinitely expandable, invent more game versions of your own. For 2 players. Includes Battle Manual and Ship Control Sheets.

For **TRS80** *Cassette*: 16K, Level II; *Disk*: 32K TRSDOS • **Apple:** *Cassette*: 16K or 32K, integer BASIC in ROM; *Disk*: 32K, integer BASIC in ROM • **Pet** *Cassette*: 8K, old or new ROMs.

$19.95 cassette • $24.95 disk

TRIPLE WARRANTY

(10) **Money back guarantee:** If you don't like the game for any reason whatever, return it intact within 10 days of receipt for a complete refund. No questions asked.

(30) **Defective warranty:** Cassette not functioning within 30 days of receipt? Return it to us and we'll exchange it. No charge, of course.

(forever) **Limited lifetime warranty:** No matter **what** happens to your cassette: the dog chewed it...you left it out in the rain...whatever. No matter **when** it happens. Return the remains to us (with $5.00 to cover all handling and shipping) and we'll send you a brand new cassette.

*We believe that computer games should be fun, challenging, intellectually stimulating...and provide you with many alternatives and ways to affect the outcome. So our games are more complex in planning your playing strategy. But **not** in the mechanics and rules of play. With all these games, **you** take command. **You** determine the course of history.*

Invasion Orion

Pit your skills against the computer! Same game system as "Starfleet Orion" but you can play it solo. 3 skill levels; the computer plays either side and takes care of the details. It has 10 fictional scenarios, 30 ship types, and 3 weapon systems.

For **TRS80** *Cassette*: 16K, Level II; *Disk*: 32K, TRSDOS • **Apple** *Cassette*: 32K Applesoft in ROM; *Disk*: 48K Applesoft in ROM • **Pet** *Cassette*: 16K, old or new ROMs.

$19.95 cassette • $24.95 disk

The Datestones of Ryn

Dunjonquest #2. Recover the datestones from the rogue Rex the Reaver and his cutthroats — who've stolen the stones from the calendar — before time runs out.
Competitive scoring system: How well can you do compared to other players?

For **TRS80** *Cassette*: 16K, Level II; *Disk*: 32K, TRSDOS • **Apple** *Cassette*: 32K Applesoft in ROM or 48K Applesoft on cassette; *Disk*: 48K Applesoft in ROM • **Pet** *Cassette*: 16K, old or new ROMs.

$14.95 cassette • $19.95 disk

Morloc's Tower

Dunjonquest #3. You'll find 3 kinds of rings, a magic sword, 2 amulets, 6 or so other treasures, 30 rooms, 18 real-time command options...and a dozen types of monsters including the heinous Morloc. Easy to learn, a challenge to master. Includes game program, 1.2 KB data file, 16 page manual.

For **TRS80** *Cassette*: 16K, Level II; *Disk*: 32K, TRSDOS • **Apple** *Cassette*: 32K Applesoft in ROM or 48K Applesoft on cassette; *Disk*: 48K Applesoft in ROM • **Pet** *Cassette*: 16K, old or new ROMs.

$14.95 cassette • $19.95 disk

Rescue at Rigel

New! Brings the Dunjonquest series to the final frontier. As Sudden Smith, with force shield and power gun, you make your way through several levels and scores of rooms to find and beam to safety the prisoners held by the evil High Tollah. Quickly, before your power pack dies and the Tollah and his minions can get to you!

For **TRS80** *Cassette*: 16K, Level II; *Disk*: 32K, TRSDOS • **Apple** *Cassette*: 32K Applesoft in ROM or 48K Applesoft on cassette; *Disk*: 48K Applesoft in ROM • **Pet** *Cassette*: 16K, old or new ROMs.

$19.95 cassette • $24.95 disk

TO ORDER:

Master Charge or Visa card holders: charge these to your credit card. Just call the appropriate toll free number: **(800) 824-7888**, operator **861**.

In California: **(800) 852-7777**, op. **861**.

In Hawaii or Alaska: **(800) 824-7919** op. **861**.

Or use the handy coupon:

Automated Simulations
Department S2
P.O. Box 4247
Mountain View, CA 94040

✓55

AUTOMATED SIMULATIONS

Please send me the following games:

Cass.	Disk	Game	
___	___	Temple of Apshai	$______
___	___	Starfleet Orion	$______
___	___	Invasion Orion	$______
___	___	Datestones of Ryn	$______
___	___	Morloc's Tower	$______
___	___	Rescue at Rigel	$______
		Plus Shipping & Handling	$ 1.00
		Sales Tax*	$______
		Total	$______

My computer is ______________________

☐ I enclose my check in the amount of ______________

☐ Please charge to my

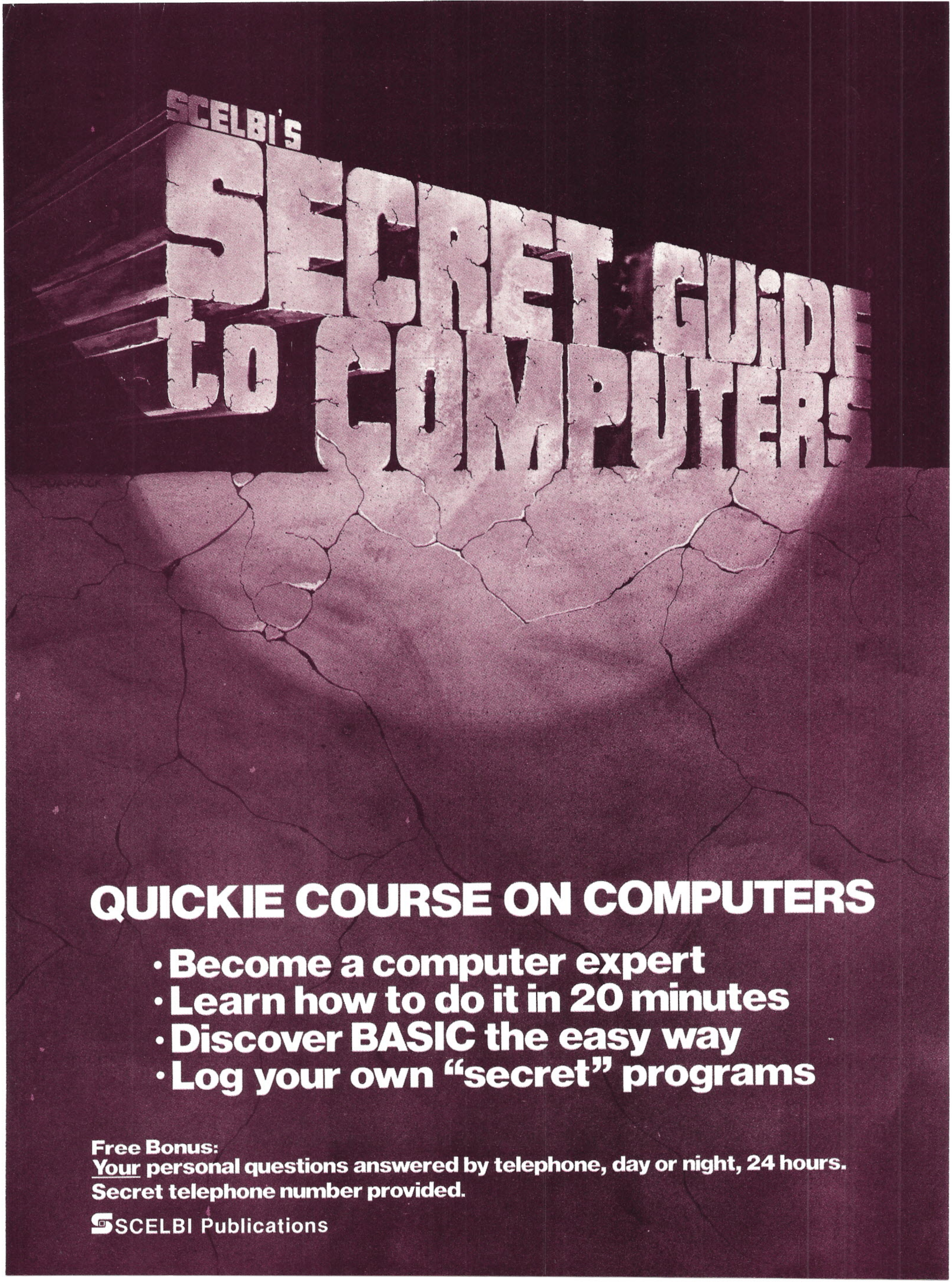
☐ Visa ☐ Master Charge:

#______________ expires ______________

Name ______________________

Address ______________________

*California residents: add 6 or 6.5% tax


SCELBI'S
SECRET GUIDE to COMPUTERS
QUICKIE COURSE ON COMPUTERS
• Become a computer expert
• Learn how to do it in 20 minutes
• Discover BASIC the easy way
• Log your own "secret" programs
Free Bonus:
Your personal questions answered by telephone, day or night, 24 hours.
Secret telephone number provided.
SCELBI Publications

Electricbill

Are you doing your bit to save energy?

Brian S. Klinger
466 Washington Road
Rye, NH 03870

What do we want from our computers? Why did we really spend the money for that exotic system that takes up space and causes us to fritter away all our free time?

For those of us who have done it, the answer is so obvious that we wonder why anyone even needs to ask. We bought our computers to have fun! And that answer satisfies us completely. But non-aficionados invariably ask two questions in this order: "How much did it cost?" and "You mean it won't do anything but play games?" These lines are usually followed by a dour shake of the head.

So we need to have our computers do useful things to satisfy our spouses, nonbelieving friends and other skeptics . . . especially when they ask how much energy the darn thing wastes!

So here is the answer. Run this program and you can not only show people a useful function of your computer, but analyze your own electricity consumption month by month. Furthermore, as anyone in New England knows, we pay more for power in this region than anyone else in the country, so conservation is vital. Anything that gives us a picture of how much energy we use will contribute to less consumption and more saving.

The Program

The program was written for my 8K PET. In addition to checking my patterns of electricity use, I wanted to accomplish two other things with the program: verify the accuracy of the electric company's calculation of my bill (even their computers have been known to drop a bit!) and learn more about the PET's ability to handle strings and graphics.

The concise computation of the monthly bill is contained in lines 800–1500. The mathematical formulas are derived from the rate sheet available from the power company, and, when rates change, it will be easy to modify lines 800 and 900 to correct the calculations.

Since I pay my monthly bill on a "budget plan" that calls for a level monthly payment of $139, line 1500 lets me know how much my payment is over/under the actual amount owed for the month.

When my PET first arrived, I studied the mimeographed instructions for using the cassette recorder to maintain data files on tape and decided this was the way to store my data. It worked fine, as long as I entered all the data at once! But figuring out a way to update the data file each month with consumption figures and the fuel adjustment charge eluded me. I could do it, but not efficiently, and I could not make the data read back fast enough. (The PET puts a long leader at the beginning of each data file, which considerably slows down search and readback of multiple files.) It seemed the only way to do it was to put the data into the program, using DATA statements. Each DATA line sets up a string (A$) for one month's figures, with the following format: date (mo/yr), *space,* kilowatt-hours (kWhr) used for general consumption, *space,* kWhr used for water heating, *space,* fuel adjustment charge.

The program then asks if you want to read back previous data. This is where you can look at your patterns of consumption to determine if your conservation measures, additional insulation, etc., have been effective or if you need to try harder. You will be asked to specify digital or graphic readback and which month's data you wish to see. Because of the size of the PET's screen, I am limited to 20 months in one run before the data begins to roll off the top of the screen. If you have a bigger screen, change lines 2025, 6025, 7025 and 8025 to whatever limit your terminal sets. If you are printing it out on a hard-copy terminal, you can eliminate the FOR I=1 TO 20-NEXT I loop.

Typing F ends the input cycle. F was not chosen arbitrarily—it stands for "finished."

Program listing.

```
0        GOSUB 10000
1-189    DATA: N = (same as line number)
           example: 34 DATA "06/78 1350 620 1.13": N = 34
190      DIM A$(N), C(N), P(N), E(N), Y$(20)
195      FOR A=1 TO N: READ A$(A): NEXT A
300      C=VAL(MID$(A$(N),7,4))
500      P=VAL(MID$(A$(N),12,3))
600      E=VAL(RIGHT$(A$(N),4))
800      F=(C-500)*.0315 + 24.55
900      G=(P-300)*.0195 + 8.52
1000     H=((C+P)/100)*E
1100     PRINT
1200     T=F+G+H
1300     T=INT(100*(T+.005))/100
1400     PRINT "TOTAL BILL IS $"; T
1500     PRINT: PRINT "BUDGET ADJUSTMENT IS $"; 139-T
1550     PRINT: INPUT "DATA READBACK (Y OR N)"; Z$
1600     IF Z$ = "Y" THEN 1650
1625     IF Z$ = "N" THEN PRINT "GOODBYE": END
1630     PRINT: PRINT "    INPUT ERROR! ! !": GOTO 1550
1650     PRINT: INPUT "DIGITAL OR GRAPHICAL (D OR G)"; Z$
1700     IF Z$ = "G" THEN 5000
1705     IF Z$ = "D" THEN 2000
```

After the column headings are printed, each Y$ you input, which is each month/year you want to see, is scanned against the A$s in memory. When a match is found, the various parts of that A$ are either printed numerically or converted to a number of dashes for the graphic plots. The program then asks if another run is wanted; if so, the program recycles.

A Few Notes

The spaces embedded in the data lines are used as delineators when reading the substrings later. If your data won't read back correctly, make sure you have put in those spaces when entering the data statements. If you change the format of the data lines to use the program for some other purpose, be sure to correct each line that calls out a substring.

The graphics used are directly from the PET's keyboard. The underlining in lines 2130, 6110, 7110 and 8110 is the shifted # character; the dashes used in the plots (lines 6140, 7140 and 8140) are the shifted C; and the X in lines 6130, 7130 and 8130 is the shifted V.

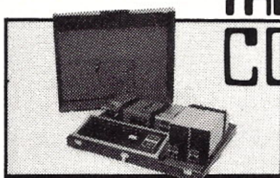
In summary, this program will allow you to keep tabs on your monthly electric bill and compare usage for any series of months desired. It also demonstrates simple techniques for manipulating strings and substrings on the PET and generating simple graphic displays of data contained in memory. I found that the PET's substring-handling capability makes this easy to do.

The program would be easy to modify for other types of data as well, and data strings could be extended to any length within the capacity of your computer. With other kinds of data requirements, it might be easier to make each data statement a shorter string and deal with it as an entity rather than breaking it up into substrings.

I hope it will also demonstrate to nonbelievers that you and your computer can do more than just play ticktacktoe or battleship.■

```
1710   PRINT: PRINT "    INPUT ERROR! ! !": GOTO 1650
2000   PRINT: PRINT "ENTER EACH MONTH/YEAR DESIRED - ENTER
       'F' FOR NO MORE DATA NEEDED"
2025   FOR I = 1 TO 20
2050   INPUT Y$(I): IF Y$(I) = "F" THEN 2100
2075   NEXT I
2100   PRINT
2120   PRINT "MONTH      DSH     QR      FUEL ADJ.CHG."
2125   PRINT "            KWH     KWH"
2130   PRINT "________________________________________"
2133   FOR W = 1 TO I
2136   FOR M = 1 TO N
2137   IF LEFT$(A$(M),5)=Y$(W) THEN C(W)=VAL(MID$(A$(M),7,4))
2138   IF LEFT$(A$(M),5)=Y$(W) THEN P(W)=VAL(MID$(A$(M),12,3))
2139   IF LEFT$(A$(M),5)=Y$(W) THEN E(W)=VAL(RIGHT$(A$(M),4))
2140   NEXT M
2141   IF Y$(W) = "F" THEN 6160
2145   PRINT Y$(W); SPC(3); C(W); SPC(3); P(W); SPC(3); E(W)
2150   NEXT W
2160   PRINT: INPUT "ANOTHER RUN (Y OR N)"; Z$
2170   IF Z$ = "Y" THEN 1650
2180   IF Z$ = "N" THEN PRINT "GOODBYE": END
2190   PRINT "    INPUT ERROR! ! !": GOTO 2160
5000   PRINT: INPUT "WHICH DATA DO YOU WANT TO SEE (D Q OR F)";Z1$
5050   IF Z1$ = "D" THEN 6000
5051   IF Z1$ = "Q" THEN 7000
5052   IF Z1$ = "F" THEN 8000
5055   PRINT: PRINT "    INPUT ERROR! ! !": GOTO 5000
6000   PRINT: PRINT "ENTER EACH MONTH/YEAR DESIRED - ENTER
       'F' FOR NO MORE DATA NEEDED
6025   FOR i = 1 TO 20
6050   INPUT Y$(I): IF Y$(I) = "F" THEN 6100
6075   NEXT I
6100   PRINT: PRINT "DSH KWH"
6110   PRINT "_______"
6120   PRINT "MONTH        KILOWATT-HOURS"
6125   PRINT "             1K    2K    3K    4K    5K"
6130   FOR J = 1 TO 40: PRINT "X";: NEXT J: PRINT
6133   FOR W = 1 TO I
6136   FOR M = 1 TO N
6137   IF LEFT$(A$(M),5)=Y$(W) THEN C(W)=VAL(MID$(A$(M),7,4))
6138   NEXT M
6140   PRINT Y$(W);: FOR R = 3 TO C(W)/150: PRINT "-";:NEXT R:PRINT
6150   NEXT W
6160   PRINT: INPUT "ANOTHER RUN (Y OR N)"; Z$
6170   IF Z$ = "Y" THEN 1650
6180   IF Z$ = "N" THEN PRINT "GOODBYE": END
6190   PRINT "    INPUT ERROR! ! !": GOTO 6160
7000   PRINT: PRINT "ENTER EACH MONTH/YEAR DESIRED - ENTER 'F'
       FOR NO MORE DATA NEEDED"
7025   FOR I = 1 TO 20
7050   INPUT Y$(I): IF Y$(I) = "F" THEN 7100
7075   NEXT I
7100   PRINT: PRINT "QR KWH"
7110   PRINT "______"
7120   PRINT "MONTH        KILOWATT-HOURS (X 100)
7125   PRINT "         1   2   3   4   5   6   7   8   9"
7130   FOR J = 1 TO 40: PRINT "X";: NEXT J: PRINT
7132   FOR W = 1 TO I
7133   FOR M = 1 TO N
7136   IF LEFT$(A$(M),5)=Y$(W) THEN P(W)=VAL(MID$(A$(M),12,3))
7138   NEXT M
7140   PRINT Y$(W);: FOR R = 6 TO P(W)/22.6: PRINT "-";:NEXT R:PRINT
7150   NEXT W
7160   PRINT: INPUT "ANOTHER RUN (Y OR N)"; Z$
7170   IF Z$ = "Y" THEN 1650
7180   IF Z$ = "N" THEN PRINT "GOODBYE": END
7190   PRINT "    INPUT ERROR! ! !": GOTO 7160
8000   PRINT: PRINT "ENTER EACH MONTH/YEAR DESIRED - ENTER
       'F' FOR NO MORE DATA NEEDED"
8025   FOR I = 1 TO 20
8050   INPUT Y$(I): IF Y$(I) = "F" THEN 8100
8075   NEXT I
8100   PRINT: PRINT "FUEL ADJUSTMENT CHARGE"
8110   PRINT "______________________"
8120   PRINT "MONTH            DOLLARS"
8125   PRINT "        .50        1.00        1.50        2.00"
8130   FOR J = 1 TO 40: PRINT "X";: NEXT J: PRINT
8133   FOR W = 1 TO I
8136   FOR M = 1 TO N
8137   IF LEFT$(A$(M),5)=Y$(W) THEN E(W)=VAL(RIGHT$(A$(M),4))
8138   NEXT M
8139   IF Y$(W) = "F" THEN 8160
8140   PRINT Y$(W);: FOR R = 9 TO E(W)*20: PRINT "-";:NEXT R:PRINT
8150   NEXT W
8160   PRINT: INPUT "ANOTHER RUN (Y OR N)";Z$
8170   IF Z$ = "Y" THEN 1650
8180   IF Z$ = "N" THEN PRINT "GOODBYE": END
8190   PRINT "INPUT ERROR! ! !": GOTO 8160
9998   REM *** "ELECTRICBILL" 07/22/78    **
9999   REM *** BY BRIAN S. KLINGER, O.D.  **
10000  PRINT CHR$(147)
10001  PRINT "TO ENTER NEW DATA, PRESS 'STOP' AND THEN"
10002  PRINT: PRINT "'LIST-150'. THEN ENTER NEW DATA AND"
10003  PRINT: PRINT "UPDATE N. THEN RUN AGAIN AND PRESS ANY"
10004  PRINT: PRINT "KEY TO CONTINUE WITH CALCULATIONS."
10005  GET S$: IF S$ = "" THEN 10005
10006  RETURN
```

THE COMP-CASE



TRS-80 COMPUTER CASE
RS201 $109

TRS-80 MONITOR CASE
RS202 $84

APPLE II CASE
AP101S Single Disk $109
AP102D Double Disk $119

APPLE II CASE
AP103M 2 Disk & Monitor $129

- PROTECTION
- PORTABILITY
- SECURITY

ATTACHE STYLE CASE FOR CARRYING AND PROTECTING YOUR APPLE OR TRS-80 COMPUTER SYSTEMS. CONSTRUCTED OF THE HIGHEST QUALITY LUGGAGE MATERIAL. WILL ACCOMMODATE COMPUTER, TAPE RECORDER OR DISK DRIVES PLUS DISKS. TAPES AND ALL MANUALS. NEVER A NEED TO REMOVE EQUIPMENT FROM CASE, SIMPLY REMOVE LID, CONNECT CABLES AND OPERATE. LID CAN BE REPLACED AND LOCKED FOR SECURITY AND PROTECTION WITHOUT DISCONNECTING CABLES.

COMPUTER CASE COMPANY ✓320

5650 INDIAN MOUND COURT
COLUMBUS, OHIO 43213
(614) 868-9464

SOFTWARE FOR OHIO SCIENTIFIC

OSI OSI

Over 80 programs for C1, C2, C4, & Superboard, on tape and disk. All come with listings and complete documentation.

GAMES-4K-Tape
CHESS FOR OSI-specify system $19.95
STARFIGHTER 5.95
Real time space war.
* ORBITAL LANDER 9.95
One or two-man piloting game.
* TIME TREK 9.95
A real time star trek game with nice graphics
ALIEN INVADERS 6.95
Rows of menacing, munching monsters march on earth.
8K GAMES
BACKGAMMON $9.95
BLACKJACK 6.95
Plays *all* Vegas rules.
Add $1.00 each for Color/Sound.

UTILITIES
C1P TEXT EDITOR $9.95
Gives real backspace, one key screen clear, *and midline editing.*
C2/4 TEXT EDITOR 9.95
Similar to C1
SUPERUTILITY 12.95
Has renumberer, Variable table maker and Search.

DISKS
SUPERDISK $24.95
(for C4MF or C8MF) contains text editor for basic lines, renumberer, new VEXEC*
THE AARDVARK/JOURNAL FOR OSI
Six issues of user information. $9.00

Our $1.00 catalog has free game and utility listings, programming hints and a lot of PEEKs and POKEs and other stuff that OSI forgot to mention—and a *lot* more programs for sale.

DISKS 5" COLOR/SOUND $29.95
DISK 1. STARFIGHTER, ROBOTANK, SEA WOLFE, BOMBER, TEN TANK BLITZ,
DISK 2 BREAKTHROUGH, LUNAR LANDER, ALIEN INVADER, KILLERROBOTS. SLASHBALL.

✓91

AARDVARK TECHNICAL SERVICES

VISA — master charge

1690 BOLTON, WALLED LAKE MI 48088 313-624-6316

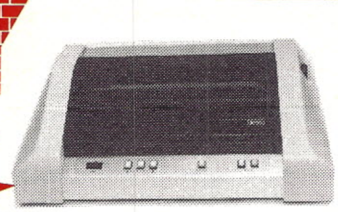
HARD COPY

AT

Soft Prices

BUY DIRECT FROM THE "Stocking SOURCE"

New!

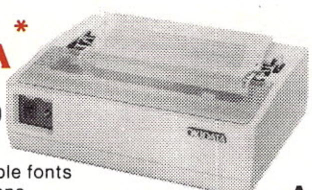


ANADEX DP-9500

Features
- 132/220 Columns
- 9 Wire Print Head
  650 Million Character Life
- 150/200 CPS
- 11x9/7x9 Character Font
- High Density Graphics
- Black Long Life Ribbon Cartridge
- Original Plus 5 Copies
- Many More Features

LIST $1650

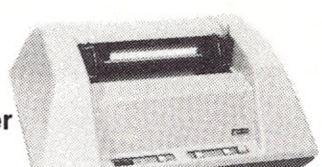
* **OKIDATA** *



Microline 80 Features
- Program-selectable fonts
- 80 and 132 columns
- 6 and 8 lines per inch
- Continuous operation—100% duty cycle
- 200,000,000 character head warranty
- 2,000,000 character ribbon warranty
- Full 96 character ASCII
- Microprocessor-controlled interfaces
- 14 pounds light
- Pin, tractor and friction feed forms
- Standard 1, 2 and 3 part paper
- Graphics

LIST ~~$945~~ $800
YOUR COST ~~$745.00~~ $600
BUY DIRECT & SAVE $200.00

* **ANADEX** *



Anadex AN Printer Model DP-8000

Features
- 80 Columns, 112 cps
- 84 Lines Per Minute, Bi-Directional, Nominal Thruput
- High Reliability—Heavy Duty Cycle, 100 M Characters Print Head Life
- Sprocket Feed
- Paper Entry Through Bottom or Rear
- 9 × 7 Character Font
- Variable Form Width

LIST $1095.00
YOUR COST 895.00
BUY DIRECT & SAVE $200.00
APPLE VERSION . . . 895.00

*

WE ARE A FACTORY AUTHORIZED FULL SERVICE CENTER

✓288

TO ORDER
CALL COLLECT
(313) 477-7586
OR
WRITE TO "THE STOCKING SOURCE"
23995 FREEWAY PARK DR.
FARMINGTON HILLS, MI 48024

Your BANKAMERICARD welcome here — VISA

BUSINESS CONTROL PROGRAMS

☐ PAYROLL ☐ GENERAL LEDGER ☐ ACCOUNTS PAYABLE ☐ ACCOUNTS RECEIVABLE

These business systems are designed with the business manager in mind. Major changes in your current bookkeeping methods are not necessary to make these programs work for you. Data may be entered into any one of the systems either directly or through subprograms, so that duplicate data entry is not required. This avoids mistakes.

Accurate reports of financial activity may be obtained on a scheduled basis or at irregular intervals.

The systems require CP/M and CBASIC. Customization, installation and training are available at additional cost. WRITE FOR SAMPLE REPORT AND TRIAL DISK—$25.00 (credited toward purchase price).

- **A Totally Linked Single Entry Accounting System**
- **Completely Screen Oriented Systems Provide User Prompting for Each Entry**
- **User Assignable Account Numbers**
- **Flexible, Reliable and Efficient**
- **Installation and Training Available**

AS LOW AS $250.00 ea.

CP/M 2.2 Control Program for Microcomputers Enabling You to Run Software Published for CP/M 1.4 on the TRS-80 Model II (Including Communications Programs)

CP/M® 2

FMG CORPORATION

CP/M is considered the industry standard disk operating system because it gives you the hardware-independent interface you need to make your computer work for you. FMG CORPORATION NOW OFFERS THE CP/M 2.2 FOR THE TRS-80 MODEL II. The latest in the evolution of a reliable and efficient software system, CP/M 2.2 features an enhanced upward compatible file system and powerful new random access capabilities. From minidisks and floppy disks, all the way to high-capacity hard disks (when available), the flexibility of CP/M 2.2 makes it a truly universal operating system. The package includes an 8" system disk, editor, assembler, debugger and communication software for the TRS-80 Model II. Special utilities not available with other CP/M systems are also included in FMG's CP/M 2.2

VISA®

(CP/M is a registered trademark of Digital Research Corp. TRS-80 is a registered trademark of Radio Shack)

$200.00

From the ORIGINATOR of the TRS-80® PROJECT

master charge THE INTERBANK CARD

THE PASCAL/MT As Low As $250.00

PASCAL/MT™, a native code PASCAL compiler designed for 8080/8085 and Z80 microcomputer systems, provides an efficient development cycle as well as efficient execution of object programs.

- Compiler executes under the CP/M operating system in as little as 32 K bytes of RAM
- Interactive Symbolic Debugger which enables the programmer to examine variables, set a breakpoint, and trace procedure calls interactively at run time
- Compiles at the rate of 600 lines per minute on a 2 MHZ 8080
- Programs Execute up to 10 TIMES FASTER than popular interpretive Pascals
- The code generated is 8080 object code which is ROMable with a minimum run time overhead of 1.5K bytes
- Interrupt procedures allow the programmer to write interrupt drivers for I/O and other real time tasks in Pascal/MT
- Bit manipulations of variables may be performed with the built-in procedures: SETBIT, CLRBIT, TSTBIT, SHL, SHR, SWAP, LO, HI.
- Assembly language subroutines may be called from Pascal/MT
- Business arithmetic version of Pascal/MT is also available
- Pascal data structures supported are: ENUMERATION AND SUBRANGE TYPES, RECORD, ARRAY, REAL, INTEGER, CHAR, and BOOLEAN
- Not implemented are: SETS, GOTO, GET, PUT

FMG CORPORATION

**P.O. Box 16020
Ft. Worth, Texas 76133
Phone (817) 294-2510**

✓70

M-443 **Send for FREE CATALOG**

PASCAL/MT™ includes compiler and a real time symbolic debugger. The system requires 32K minimum and 2 mini disks or one 8" disk.

Superboard II: A Second Look

A year later, it's time for another look at OSI's Superboard II.

Rich Jensen
7612 Haymarket Lane
Raleigh, NC 27609

A July 1979 *Microcomputing* article by Bruce Chamberlain about the OSI Superboard II ("OSI's Superboard II," p. 66) provided a good introduction to that 6502-based system: 4K of RAM, 8K ROM with Microsoft BASIC, 2K ROM monitor, 1K of display buffer RAM, video interface, audio interface and full ASCII keyboard, wired and tested, all for $279. For $350, you get the whole thing in a metal case with a power supply. The combination is called the Challenger 1P.

The Superboard met my requirements for hardware and software experimentation. This article describes two hardware modifications I made; it also documents the results of digging useful subroutines and information out of the available documentation and the ROM firmware.

The Hardware

The circuit board is a 10 × 14 inch double-sided board that is prewired for an RS-232 interface, an external 40-pin bus (OSI standard), an audio output and joystick connection (not documented), several unused prototyping sockets and connectors, plus a few goodies I haven't figured out yet.

Experimenting with the electronics is easy. I have modified the video output to work with a Ball monitor and added a switchable tape rate selector for 600 cps, twice the initial speed. I made both modifications on the existing circuit board. Fig. 1 shows the circuit modifications, and Photo 1 shows the parts placement.

The video modifications provide a longer vertical sync pulse, caused by increasing c8 to .2 uF, and an inverted horizontal sync during vertical sync time—this is the reason for rewiring U70 & U65. Notice that I modified U65 at the socket by soldering small wires on pins 4 and 13 and plugging them into the other socket connections to keep the board changes to a minimum. I split the connection between pins 9, 10 and 11 with an X-acto knife and rewired as shown with fine wire soldered directly to the board. This change requires no more parts than the original circuit and works fine with a normal TV also.

Function	Address hex	Address decimal	Description
scroll screen	BF2D	48941	This is the BASIC print routine. It checks the contents of the accumulator for carriage return, line feed or valid character. It will display the character at the next available position at the bottom line of the screen and scroll all lines up as needed (just as PRINT statements in BASIC). Load accumulator with the character to be displayed and use the JSR instruction 20 2D BF.
initialize cassette interface	FCA6	64678	This routine writes a 3H, 11H to the cassette interface ACIA at address F000H. This resets the ACIA and initializes if for the proper mode and clock division ratio. Use a JSR 20 A6 FC once in your program before you read or write from the tape.
write to cassette	FCB1	64689	This routine writes the contents of the accumulator to the cassette. It checks for write complete and will not return until ready for another character. You can write characters by loading them into the accumulator and using JSR 20 B1 FC without regard to overrunning the interface.
read ASCII from cassette	FE80	65152	This routine reads ASCII data from cassette and turns off high-order bit. Issue JSR 20 80 FE. Character read will be placed in the accumulator. This routine is also self-timing.
convert ASCII to binary	FE93	65171	This routine takes ASCII characters 0-1 and A-F in the accumulator and returns the binary values 0-15. Use JSR 20 93 FE.
write to screen and cassette	FF69	65385	This routine writes to both the screen—it calls the BASIC print routine—and if the contents of the save flag (205H) are nonzero, it writes to the cassette. When it encounters a carriage return it includes 10 nulls after each carriage return. Use JSR 20 69 FF.

With the above routines you can start writing full machine-language programs that consist mostly of JSRs to the existing monitor code.

Table 1.

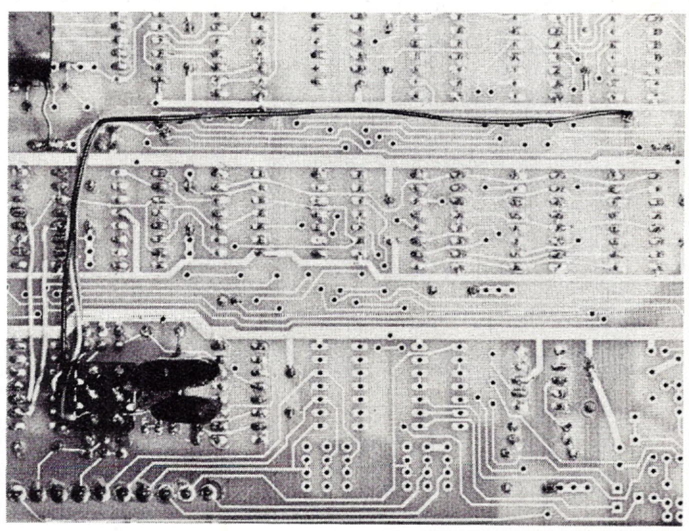


Photo 1. Parts placement.

This modification is only required for a monitor without internal sync oscillators. A normal TV will work fine unmodified with a video converter. A external circuit to do the same thing is described in the original reference.

The cassette speed modification is simple and worthwhile if you become impatient loading an 8K program from the tape. I broke down and bought the 4K of additional RAM that is presocketed on the board. The switch selects the next higher clock signal from the master divider chain to provide a 16 × clock of 9600 Hz rather than 4800 Hz. Wire is routed from the nearest point on the board that provides the C3 signal. This signal goes to a double-pole, double-throw switch which selects either C3 or C4 while selecting the proper capacitor to provide receive timing. The modification doubles the maximum audio frequency to 4800 Hz on the tape instead of the 2400 Hz now used.

CX halves the timing rate of the one-shot U69 to decode the new signal properly on receive. I have had no error rate problems with my low priced Sankyo recorder and cheap tape. I suggest you resist the temptation to simplify the circuit any further by eliminating the switch-selectable 300 Hz rate. Someday you may want to buy a prerecorded program which has been loaded at that rate. Someone with a little soldering experience can easily make the tape modification. The video interface requires good eyesight, and a steady hand to cut and resolder the connections on IC U70.

The Software

Being of the "old school," I wanted to write and execute machine-language code. This is almost always the fastest way to do things; sometimes it is the only way. The 65 V ROM monitor that comes with the system contains functions to allow you to display and modify the contents of RAM storage directly from the keyboard. You key in a hex address and the contents are displayed in hex on the monitor. It also provides functions to load a machine-language program in the special monitor format from the audio cassette. The documentation does not tell you how to create the tape, however! See Fig. 2 for a BASIC program to create a monitor input tape.

My first weeks on the system were devoted to using the monitor to display itself and laboriously writing down and decoding all 2000 bytes. This brought me a wealth of usable subroutines that could be called from BASIC or machine-language programs. These routines are debugged, take no additional storage and are always in the system. I've documented some of the most valuable routines and included their location addresses along with some examples of their use.

Problems and Solutions

You will probably need to know how to use machine-language subroutines from a BASIC program. The first problem is entering the machine-language code into the system, if it's not already there. Fig. 3, lines 1–3, shows a BASIC routine to read a data statement and poke the contents into a series of storage locations where it will run.

This is the easiest approach for a routine you want to store

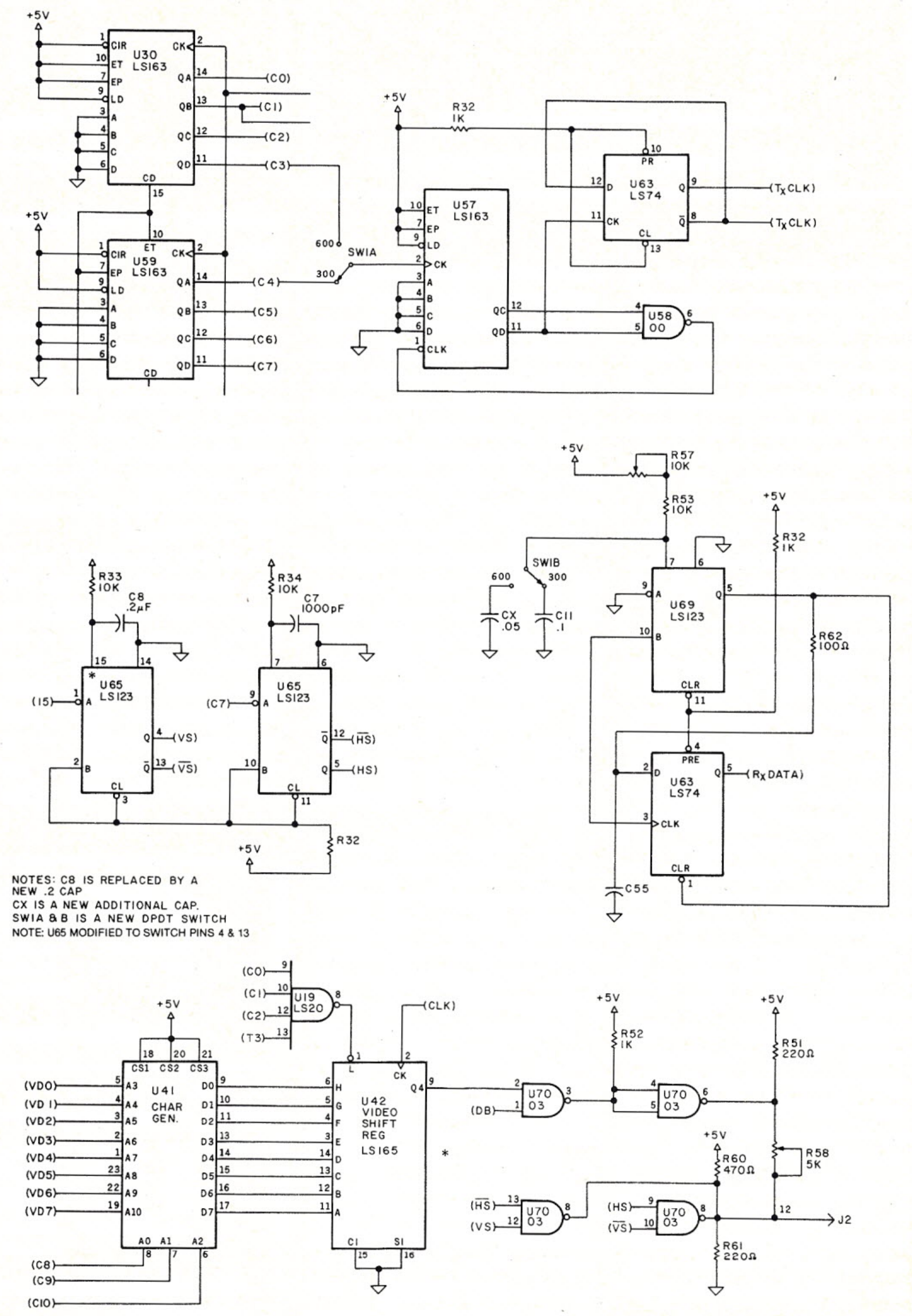


Fig. 1. Circuit modifications.

```
5 DIMA A$(255)
10 FOR X=1 TO 255
20 INPUT A$(X)
30 IF A$(X)="XXX" GOTO 41
40 NEXT
41 SAVE
42 PRINT "TURN ON TAPE RECORDER TO RECORD"
43 INPUT "HIT ANY KEY TO CONTINUE";B$
44 FOR Y=1 TO 5000:NEXTY
45 X=X-1
46 FOR W=1 TO X
47 PRINT A$(W)
49 NEXTW
50 LOAD
55 PRINT "SAVE COMPLETE"
60 END
RUN
 ? .0222
 ? /20
 ? 05
 ? AE
 ? etc
 ? XXX
will load 0222 with 20 05 AE etc under monitor control
```

Fig. 2. BASIC program to create monitor format tape.

```
1 DATA 32,5,174,164,175,200,165,174,32,193,175,96
2 REM X20,05,AE,A4,AF,C8,A5,AE,20,C1,AF,60
3 FOR X=0 TO 11:READ Y:POKE 546+X,Y:NEXTX
4 REM loads subroutine into 222 to 22D
10 POKE 11,34:POKE 12,2
12 REM poke B with 22, poke C with 2
20 FOR B=1 TO 10
30 A=USR(B)
40 PRINT B,A
50 NEXTB

RUN
         1       2
         2       3
         3       4
         4       5
         5       6
         6       7
         7       8
         8       9
         9      10
        10      11
```

Fig. 3. Example of USR coding.

with the main program. The secret is to convert the hexadecimal machine codes to decimal data for use with POKE statements. The elegant way is to write a BASIC program that converts hexadecimal to decimal and use the program for conversion. The alternate way is to look it up one byte at a time in a conversion table. You may want to use the monitor facility to key in the subroutine directly the first time for testing.

Now that you have the program in storage you must set up pointers to the program address and then execute a USR(X) statement in BASIC to cause a jump-to-subroutine (JSR) instruction to occur. The question is, where does it jump to? The OSI reference manual says you must load address 23EH and 23FH with the address of your subroutine. This doesn't work on my system, but loading address BH and CH does. Line 10 of Fig. 3 pokes this location with the decimal address of the subroutine. You or the program must do this before the USR is issued. You may pass a parameter to the subroutine as the value of X in the USR(X) statement. This value, in turn, passes to the subroutine in storage locations AFH and AEH if it calls the routine pointed to by 6H and 7H (AE05H in line 222 of Fig. 4).

The subroutine can pass a value back to the basic program by calling the routine pointed to by 8H and 9H (AFC1H). The value will return to BASIC as the value of AH in the statement A=USR(X). For multiple values or as an alternate technique, you can pass value back to BASIC by setting a fixed storage location and examining the contents with a PEEK statement.

This sounds complicated, but by using the system manuals and Fig. 4, you should be able to work out what is going on. The example passes a number to the subroutine, which adds one to the number and passes the result back. Not very sophisticated, but it proves the linkage works, and you can go on from there.

Next, decide where the subroutine is located in storage. If you are running BASIC with the machine code as a subroutine, you normally reserve an area in high core large enough for the subroutine to fit. The basic program loads into memory starting at address 0300H to high memory. You indicate to the start-up routines how much storage you want BASIC to use by responding to the D/W/C/M ? with C. Then answer "memory size?" with a value that equals your user RAM (4096 in the initial system) minus the number of bytes you want to set aside. The problem is that you will have to do this every time you cold-start the system and run this program.

A Trick

Hex locations 0222 to 02FA are unused by either BASIC or the system code. You can fit 216-byte subroutines into this area and not have to answer properly every time you want to use the program. Fig. 3 has an example of a subroutine loaded into this free area.

Now that you know how to use subroutines with the OSI machine, look at some that are already coded for you in the monitor and BASIC ROM. Located at FEEDH is a routine to read ASCII characters directly from the keyboard. See Fig. 5 for an example. Using the routine this way in lieu of an INPUT statement, you can enter data into the program without causing the screen to scroll and ruin the graphics display. This example also shows use of the ASCII-to-binary conversion subroutine at FE93H. Without the JSR to this routine you get the ASCII character values for the key pressed: with it you get the numeric values 1 to 15.

Another method explained in the OSI manuals is to do a PEEK at decimal 57088 (the keyboard address) and a whole series of IF statements to decode the non-ASCII results. This uses more storage space and execution time that the machine-level subroutine does. Table 1 contains a list of some subroutines, with their addresses and a description of their function. ■

```
222  20 05 AE   jump to subroutine pointed to by 6,7 AE05
225  A4 AF      LDYZP get low order parameter from 00AF
227  C8         INY add 1
228  A5 AE      LDAZP get high order parameter from 00AE
230  20 C1 AF   JSR to subroutine pointed to by 8,9 AFC1
233  60         RTS return to BASIC program
```

Fig. 4. Machine-language subroutine used above.

```
222  20 ED FE   JSR to FEED "read ASCII"
225  20 93 FE   JSR to FE93 "convert ASCII to binary"
228  A8         TAY move char to Y reg to return to BASIC
229  A9 00      LDAIM 0 A reg
22B  20 C1 AF   JSR routine to return char to BASIC
22E  60         RTS return
```

Fig. 5. Machine-language subroutine to read binary numbers from keyboard.

SORTIT: A Sort Program

SORTIT enhances two of the author's previous programs: SCREEN and FILEIT.

Forest E. Myers
5114 Garnett St.
Shawnee, KS 66203

Several months ago I introduced two programs to create, manipulate and retrieve data files. The first program, titled SCREEN, appeared in the April 1980 issue. This program was used to define data fields within a record. The second program, called FILEIT (May 1980 issue), created, modified and retrieved information from data files. The purpose of this article is to introduce another program, SORTIT, which enhances the usefulness of the above programs.

SORTIT is a multikeyed sort program designed to sort data files generated by FILEIT. As was the case with SCREEN and FILEIT, SORTIT is written in Business BASIC, an interpreter sold by Microworks of Cincinnati OH.

Sort Levels

SORTIT supports up to five levels of sort in either ascending or descending order. The order of sort, however, must be the same at all levels. This means that higher-level sorts cannot be sorted one way, for example, ascending order, while lower-level sorts are sorted in another, for example, descending order.

The best way to explain the sort level concept is by an example. Assume you are working in the personnel department of a small company. On the company's microcomputer you have developed a personnel file using a version of FILEIT. In the personnel file you created, you have captured information on employee name, address, department, job title, home phone, sex, marital status, children, date of employment, previous employer and current office telephone number.

One day you walk into the office, where an employee is groaning over the prospects of the annual effort to put out the company telephone directory. Although there are only 100 employees in the company, they are forever changing offices and telephone numbers. It is always a mess to keep things in order, and a last-minute update causes havoc.

After listening to this, you think of your personnel file and the information kept in it. You then suggest that there might be a way to ease the employee's pain: use the information in the personnel file. Just reorder the file so a directory can be generated from it.

To generate the directory, you decide a two-level sort is needed. The first sort level, or key, is the employee's last name. The second sort level is the employee's first name. Once sorted using this scheme, the file will contain the names of the employees in alphabetical order by last name, and within each last name employees will be ordered alphabetically by first name. After the file is sorted, the file retrieval option of FILEIT can be used to print out the newly sorted file.

Although the example presented was originally intended to describe the concept of sort levels, it also demonstrates how files intended for one purpose may be used to meet others. SORTIT, therefore, offers the ability to extend file use. As such, it offers savings in time and money.

Operation

When SORTIT is first run, it asks for the number of sort levels, or keys, to be used. A number between one and five must be entered. After the number is entered, SORTIT asks for the order of sort. If the file is to be sorted in ascending order, a "one" is entered. Entering a "two" results in the file being sorted in descending order. SORTIT then reminds the user that a number between one and the number of keys specified be placed after the requisite number of data fields serving as sort criteria.

As a prompting device for sort level specification, SORTIT presents a FILEIT data entry screen. Recall that this screen consists of labels identifying data fields within a record and a series of small x's after each label showing the specified length of each data field. Instead of entering data after each label, the user enters a single digit, from one to five, after the labels to serve as sort criteria. If a given label is not to serve as a sort criterion, a carriage return is entered in response to the prompting question mark. This will move the prompt behind the next label on the screen.

Once all sort keys have been entered, SORTIT asks for the name to be given the sorted file. After the name is entered, SORTIT reads through the file to be sorted. From each record, SORTIT extracts the fields serv-

Program listing.

```
10  DIM A$(256),B$(10),C$(10),E$(512),C(60)
20  DIM R(60),L(60),F$(256)
30  DIM G$(256),T$(10),B(60)
40  DIM D(60)
50  #"" : FOR I=1 TO 7 : #" " : NEXT I
60  #"                              SORT PROGRAM"
70  #"                                08/14/79"
80  REM  INITIALIZE STRINGS BEFORE FIRST USE -- A REQUIREMENT
90  REM  OF BUSINESS BASIC
100  A$=" " : FOR I=1 TO 8 : A$=A$+A$ : NEXT I
110  B$=A$(1,10) : C$=B$ : E$=A$+A$ : F$=A$
120  G$=A$ : T$=C$
130  FOR I=1 TO 5 : #" " : NEXT I
140  INPUT "Enter name file to be sorted ",B$
150  IF LEN(B$)<>6 THEN #"Only six characters are allowed " : GOTO 140
```

ing as sort criteria and places them according to the order specified by the user in sort string S$.

The data field given a value "1" by the user will be put into the sort string first. The field given a value of "2" will be placed in the sort string next and so on. The effect of this is to make the data field given a value "1" the primary sort field, such as employee's last name was in the example. Each higher numerical value specifies another sort level. In the example, the employee's first name would have been given a value "2," resulting in the first names' being sorted in order within each last name, e.g., Myers Forest, Myers Jaime, Myers Jennifer and so on.

By extracting only that portion of the record used for sorting purposes, you can save memory space. At the time of extraction, the block and record from which the sort data were extracted are also placed in the sort string. This allows retrieval of the entire record once the sort is completed. Once all records have been read, SORTIT reorders the sort string according to the specified sort criteria.

When the sort is completed, SORTIT then uses the previously stored block and record addresses to read in the entire record. This information is then output to the file specified as the sorted file. It should be noted that SORTIT will not output a record until a data block is filled, then the whole block is output. This saves diskette space.

When SORTIT completes the sorting process, a sorted file will be created. It will be identical to the original file, except ordered according to user specifications.

The source listing for SORTIT is included. Remark statements have been provided to act as a guide. I hope these will ease the burden of following the program.

In summary, I wrote SORTIT to enhance the capabilities of FILEIT. By providing the facility for sorting FILEIT data files, the usefulness of these files has been extended. ■

```
160  B$=B$+".DA"+CHR$(0)
170  INPUT "Enter disk drive number where file can be found ",D
180  IF 0<0 OR D>1 THEN #"Only 0 or 1 is allowed " : GOTO 170
190  INPUT "Enter name to be given sorted file ",T$
200  IF LEN(T$)<>6 THEN #"Only six characters allowed " : GOTO 190
210  T$=T$+".DA"+CHR$(0)
220  IF T$=B$ THEN #"Sorted file name must be different than source file name. "
230  IF T$=B$ THEN 190
240  INPUT "Enter disk drive number where sorted file is to be written ",D0
250  IF D0<0 OR D0>1 THEN #"Only 0 or 1 is allowed" : GOTO 240
260  REM THE FORMAT FOR THE OPEN STATEMENT IS A AS FOLLOWS
270  REM     THE 2 REPRESENTS THE LOGICAL FILE NUMBER.  FURTHER
280  REM     REFERENCES TO THE FILE FOR I/O ARE DONE THROUGH THIS
290  REM     NUMBER.
300  REM     THE "E" IS A RETURN CODE VARIABLE.  EXAMINATION OF
310  REM     THIS VARIABLE LETS YOU KNOW IF I/O OPERATIONS TO THE
320  REM     DISK WERE CARRIED OUT CORRECTLY.
330  REM     B$ IS THE NAME GIVEN THE FILE PLUS ANY APPENDED FILE
340  REM     EXTENSION.  A CHR$(0) IS REQUIRED TO BE ADDED.
350  REM     THE 2 AFTER THE B$ REPRESENTS THE TYPE OF FILE OPERATION
360  REM     TO BE DONE.  A "1" IS AN OUTPUT OPERATION ONLY.  A
370  REM     "2" IS AN INPUT OPERATION ONLY.  A "3" IS INPUT AND
380  REM     OUTPUT OPERATION TO DISK.
390  REM     THE D REPRESENTS THE DRIVE NUMBER WHERE THE SPECIFIED
400  REM     FILES MAY BE FOUND.
410  REM     ANOTHER VARIABLE MAY BE ADDED AFTER THE DRIVE DESIGNATOR.
420  REM     THIS VARIABLE IS USED TO INDICATE THE NUMBER OF BLOCKS
430  REM     OF DISKETTE SPACE THE FILE WILL REQUIRE.
440  OPEN (1,E,B$,2,D)
450  IF E>=1 THEN 2370
460  GET (1,E,A$,0)
470  IF E>=1 THEN 2370
480  C$=A$(241,242) : CONVERT C$ TO Z : REM Z=NUMBER OF LABELS
490  C$=A$(243,244) : CONVERT C$ TO L1 : REM L1=1ST SCREEN LINE
500  C$=A$(245,246) : CONVERT C$ TO B1 : REM RECORDS PER BLOCK
510  C$=A$(247,249) : CONVERT C$ TO D : REM D=LENGTH OF RECORD
520  C$=A$(250,252) : CONVERT C$ TO Z3 : REM Z3=NUMBER OF RECORDS
530  C$=A$(253,255) : CONVERT C$ TO R : REM R=MAX NUMBER OF RECORDS
540  REM FIGURE NO. OF BLOCKS NEEDED FOR SORTED FILE.  IF A>0
550  REM THEN A PARTIAL BLOCK IS NEEDED THEREFORE BUMP BLOCK
560  REM COUNTER.
570  I2=INT(R/B1) : A=R-I2*B1 : IF A>0 THEN I2=I2+1
580  REM I2 IS BLOCK COUNTER.  4 BLOCKS ARE ADDED FOR HOUSEKEEPING
590  REM BLOCKS PLACED IN FRONT OF THE FILE.
600  I2=I2+4
610  OPEN (2,E,T$,3,D0,I2)
620  FOR I=1 TO Z
630    C$=A$(I*2-1,I*2) : CONVERT C$ TO L(I) : REM LABEL LENGHTS
640    C$=A$(119+I*2,120+I*2) : CONVERT C$ TO D(I) : REM DATA FIELD LENGTHS
650    NEXT I
660  REM READ IN LABELS AND PUT THEM IN E$.
670  FOR I=1 TO 2
680    GET (1,E,A$,I)
690    IF E>0 THEN 2370
700    E$(I*256-255,I*256)=A$
710    NEXT I
720  GET (1,E,A$,3)
730  IF E>0 THEN 2370
740  FOR I=1 TO Z
750    C$=A$(I*2-1,I*2) : CONVERT C$ TO R(I) : REM CURSOR ROW ADDRESSES
760    C$=A$(119+I*2,120+I*2) : CONVERT C$ TO C(I) : REM CURSOR COLUMNS
770    NEXT I
780  REM SORT KEY SPECIFICATION ROUTINE
790  #""
800  FOR I=1 TO 5 : #" " : NEXT I
810  INPUT "Enter number of sort keys (5 maximum) ",K0
820  IF K0>5 THEN #"5 is maximum sort keys allowed " : GOTO 810
830  INPUT "Order of sort  1. Ascending  2. Descending ",O0
840  #"Place a number 1 thru ";K0;" after labels that are to serve"
850  #"as sort keys. "
860  #" " : INPUT "Press return to continue ",C$
870  GOSUB 2740
880  #"Enter sort key number at ? above ";
890  K9=0
900  FOR I1=1 TO Z
910    REM POSITION PROMPTING ? ONE SPACE AFTER LABEL.
920    CURSOR R(I1),C(I1)+L(I1)
930    REM HAVE USER INPUT SORT KEY NUMBER. A VALUE BETWEEN 1 AND 5.
940    REM A STRING VARIABLE IS USED TO ALLOW ENTRY OF A CARRIAGE
950    REM RETURN TO REPOSITION ? BEHIND NEXT LABEL.
960    G$=F$ : REM NULL OUT G$. NOTE F$ IS USED ONLY TO NULL OTHER STRINGS
970    INPUT1 G$(I1,I1)
980    #" ";
990    REM BLANK OUT X'S AFTER NUMBER OR CR ENTERED
1000    Z2=D(I1)-1 : FOR Z5=1 TO Z2 : #" "; : NEXT Z5
1010    REM THE NEXT LINE GOES BACK AND BLANKS OUT THE LEFT OVER
1020    REM PROMPTING ?
1030    CURSOR R(I1),C(I1)+L(I1) : #" ";
1040    IF G$(I1,I1)=" " THEN 1150
1050    C$=G$(I1,I1) : CONVERT C$ TO A0
1060    K9=K9+1
1070    REM COUNT THE NUMBER OF SORT KEYS ENTERED (K9)
1080    REM PLACE SORT KEY VALUE ENTERED BY USER FOR LABEL INTO O()
```

```
1090     REM PLACE LABEL'S RELATIVE POSITION INTO V().  WITH THIS
1100     REM SCHEME, THE LABELS ON THE SCREEN CAN BE GIVEN ANY VALUE
1110     REM FROM 1 TO 5 WITHOUT CONCERN ABOUT THE ORDER OF LABEL
1120     REM APPEARANCE ON THE SCREEN.
1130     O(K9)=A0 : V(K9)=I1
1140     IF K9=K0 THEN EXIT 1230
1150     NEXT I1
1160   CURSOR 15,0
1170   #"                                                      ";
1180   CURSOR 15,0
1190   IF K9<K0 THEN #"Not enough sort keys specified "
1200   FOR C0=1 TO 500 : # : NEXT C0
1210   GOTO 780
1220   REM
1230   GOSUB 2630 : REM PUT THE SPECIFIED SORT KEYS INTO ORDER
1240   G$=F$
1250   REM DETERMINE THE TOTAL LENGTH OF THE SORT KEYS SPECIFIED.
1260   FOR I=1 TO K0
1270     U=U+D(V(I))
1280     NEXT I
1290   REM ADD SIX TO TOTAL SORT KEY LENGTH (U) FOR BLOCK AND RECORD
1300   REM DESIGNATOR.
1310   U0=U+6
1320   U1=U0*Z3
1330   IF U1>2048 THEN #"Not enough room to sort file " : END
1340   FOR I=1 TO K0
1350     D3=0
1360     REM CALCULATE THE BEGINNING AND ENDING POSITION FOR EACH
1370     REM DATA FIELD SPECIFIED BY THE USER AS A SORT KEY.  THE
1380     REM DATA FIELD WAS SPECIFIED WHEN THE USER PLACED A NUMBER
1390     REM BEHIND A LABEL ON THE DATA ENTRY SCREEN.
1400     FOR J=1 TO V(I)
1410       D3=D3+D(J)
1420       NEXT J
1430     B0(I)=D3-D(J-1)+1
1440     E0(I)=D3
1450     NEXT I
1460   REM INITALIZE SPACE ON THE DISKETTE FOR THE SORTED FILE
1470   REM CLEAR SCREEN, HOME CURSOR AND DISPLAY MESSAGE
1480   #"" : #"Blanking out diskette for sorted file "
1490   FOR Z0=1 TO I2
1500     PUT (2,E,F$,Z0)
1510     NEXT Z0
1520   REM TRANSFER HOUSEKEEPING BLOCKS TO FRONT END OF SORTED FILE
1530   FOR Z0=0 TO 3
1540     GET (1,E,A$,Z0)
1550     PUT (2,E,A$,Z0)
1560     NEXT Z0
1570   UNDIM E$,C,R,L,B,R8,L8,T3,F
1580   REM S$ IS REFERRED TO AS THE SORT STRING.  IT IS INTO THIS
1590   REM STRING THAT THE VALUES FOR THE SORT KEYS ARE TAKEN FROM
1600   REM EACH RECORD AND PLACED FOR SORTING.
1610   DIM S$(2048)
1620   S$=" " : FOR I=1 TO 11 : S$=S$+S$ : NEXT I
1630   Z4=0
1640   FOR B=4 TO I2
1650     GET (1,E,A$,B)
1660     IF E>1 THEN #" GET ERROR " : STOP
1670     FOR J=1 TO B1
1680       Z4=Z4+1 : IF Z4=Z3+1 THEN EXIT 2030
1690       X6=0
1700       REM Z4 SERVES AS A RECORD COUNTER
1710       REM X8 HOLDS THE BEGINNING POSITION OF FIELD SERVING AS SORT KEY
1720       REM X7 HOLDS THE ENDING POSITION OF FIELD SERVING AS SORT KEY
1730       REM X6 HOLDS THE SUM OF THE SORT KEY LENGTHS
1740       REM X AND Y ARE USED TO POSITION SORT DATA IN THE SORT STRING.
1750       REM K0 HOLDS THE NUMBER OF SPECIFIED SORT EKYS
1760       REM  WITH EACH PASS THROUGH THE LOOP, THE SORT KEY INFORMATION
1770       REM IS EXTRACTED FROM THE DATA RECORD AND PLACED IN THE SORT
1780       REM STRING.  THE SORT STRING IS THEN SORTED.
1790       REM BY USING THIS APPROACH ONLY THE SORT KEY ARE IN MEMORY.
1800       REM THIS ALLOWS SORTING A MUCH LAGER DATA FILE WITHOUT HAVING
1810       REM TO SEGMENT IT.
1820       FOR I=1 TO K0
1830         X=Z4*U0-U0
1840         Y=E0(I)-B0(I)
1850         X8=J*D-D+B0(I)
1860         X7=J*D-D+E0(I)
1870         X6=X6+E0(I)-B0(I)+1
1880         REM PLACE PORTION OF RECORD SPECIFIED AS SORT KEYS INTO SORT
1890         REM STRING.
1900         S$(X+X6-Y,X+X6)=A$(X8,X7)
1910         NEXT I
1920       REM PUT BLOCK AND RECORD IDENTIFIERS INTO SORT STRING
1930       REM ONCE SORT STRING IS SORTED INTO ORDER, THE BLOCK AND
1940       REM RECORD INFORMATION WILL BE USED TO GET ENTIRE RECORD
1950       REM FROM SOURCE FILE AND WRITE IT OUT TO SORTED FILE.
1960       CONVERT B TO C$(###)
1970       S$(Z4*U0-5,Z4*U0-3)=C$
1980       CONVERT J TO C$(###)
1990       S$(Z4*U0-2,Z4*U0)=C$
2000       NEXT J
2010     NEXT B
```

MORE™
EPROM PROGRAMMER

- 3K RAM EXPANSION SPACE
- OUTPUT PORT EXPANSION
- EPROM SOCKET FOR OFTEN NEEDED SOFTWARE
- READY TO USE ON BARE

KIM, SYM, AIM

BOARD, SOFTWARE ON KIM FORMAT TAPE, MANUAL, LISTINGS, ALL PERSONALITY KEYS FOR 2708, 2716 (±5 +12V) AND 2716, 2758, TMS 2516 (5V ONLY) -- $169.95

- 2708 EPROM WITH SOFTWARE IS $20.00

T.T.I. P.O. Box 2328 Cookeville, TN 38501
Phone: 615-526-7579 ✓204

EXIDY Sorcerer 16K $1,089*
commodore Pet 16K $799*

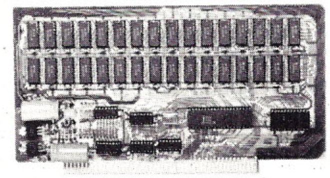
Call or write for price list with comparable savings on 32/48K models and peripherals. ✓260

MicroComputer Warehouse
3620 La Habra Way
Sacramento, Ca. 95825
(916) 486-3678

- Computers are fully tested and burned in for 48 hours.
- We respond promptly to all inquiries and orders. Send $1 for catalog, refundable with order.

*Cash price-add 2% for Visa/Mastercharge. Price subject to change without notice.

The days of complicated, unreliable, dynamic RAM are gone:



INTRODUCING
JAWS
the ultrabyte memory board

$199.95 (*complete kit with 16K memory*)

Netronics consistently offers innovative products at unbeatable prices. And here we go again — with JAWS, the ultrabyte 64K S100 memory board.

ONE CHIP DOES IT ALL

JAWS solves the problems of dynamic RAM with a state-of-the-art chip from Intel that does it *all*. Intel's single chip 64K dynamic RAM controller eliminates high-current logic parts . . . delay lines . . . massive heat sinks . . . unreliable trick circuits.

REMARKABLE FEATURES OF JAWS

Look what JAWS offers you: Hidden refresh . . . fast performance . . . low power consumption . . . latched data outputs . . . 200 NS 4116 RAMs . . . on-board crystal . . . 8K bank selectable . . . fully socketed . . . solder mask on both sides of board . . . designed for 8080, 8085, and Z80 bus signals . . . works in Explorer, Sol, Horizon, as well as all other well-designed S100 computers.

GIVE YOUR COMPUTER A BIG BYTE OF MEMORY POWER WITH JAWS — SAVE UP TO $90 ON INTRODUCTORY LIMITED-OFFER SPECIAL PRICES!

UNDECIDED? *TRY A WIRED 16K JAWS IN YOUR COMPUTER ON OUR 10-DAY MONEY-BACK OFFER (SPECIFY YOUR COMPUTER).*

CONTINENTAL U.S.A. CREDIT CARD BUYERS OUTSIDE CONNECTICUT CALL
CALL TOLL FREE 800-243-7428
From Connecticut Or For Assistance, (203) 354-9375

NETRONICS RESEARCH & DEVELOPMENT LTD.
333 Litchfield Road, New Milford, CT 06776

Please send the items checked below: Dept. K-7

- ☐ **JAWS 16K RAM kit**, No. 6416, $199.95.*
- ☐ **JAWS 16K RAM fully assembled, tested, burned in**, No. 6416W, $229.95.*
- ☐ **JAWS 32K RAM kit**, No. 6432, (reg. price $329.95), **SPECIAL PRICE $299.95.***
- ☐ **JAWS 32K RAM fully assembled, tested, burned in**, No. 6432W, (reg. price $369.95), **SPECIAL PRICE $339.95.***
- ☐ **JAWS 48K RAM kit**, No. 6448, (reg. price $459.95), **SPECIAL PRICE $399.95.***
- ☐ **JAWS 48K fully assembled, tested, burned in**, No. 6448W, (reg. price $509.95), **SPECIAL PRICE $449.95.***
- ☐ **JAWS 64K RAM kit**, No. 6464, (reg. price $589.95), **SPECIAL PRICE $499.95.***
- ☐ **JAWS 64K RAM fully assembled, tested, burned in**, No. 6464W, (reg. price $649.95), **SPECIAL PRICE $559.95.***
- ☐ **Expansion kit, JAWS 16K RAM module**, to expand any of the above in 16K blocks up to 64K, No. 16EXP, $129.95.*

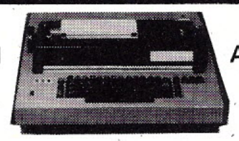
*All prices plus $2 postage and handling. Connecticut residents add sales tax.

Total enclosed: $ ________

☐ Personal Check ☐ Money order or Cashiers Check
☐ VISA ☐ MASTER CHARGE (Bank No. ____)

Acct. No. ________ Exp. Date ____
Signature ________
Print Name ________
Address ________
City ________
State ________ Zip ________
☐ Send me more information

SURPLUS ELECTRONICS



IBM SELECTRIC BASED I/O TERMINAL WITH ASCII CONVERSION INSTALLED $645.00

- Tape Drives • Cable
- Cassette Drives • Wire
- Power Supplies 12V15A, 12V25A, 5V35A Others, • Displays
- Cabinets • XFMRS • Heat Sinks • Printers • Components

Many other items. SEND $1.00 FOR CATALOG. REFUNDABLE FIRST ORDER

WORLDWIDE ELECT. INC.
130 Northeastern Blvd.
Nashua, NH 03060 ✓122

Phone orders accepted using VISA or Master Charge
call 603-889-7661

CUDDLY SOFTWARE

Unbreakable Systems/Support Software for any 1802 system.

CSOS Series (Operating System) — aids development and modification of user programs with software tools, like Add/Delete Byte, which also expand/contract the remainder of the memory page. User Programs also interface via Std. Call/Ret. for video routines, I/O drivers, etc. 1861 alphanumeric output via Screen Driver (10 lines, ave. 17 char/line, auto-scroll). User graphics also made easy.

CSTP Series (Trace Program) — 1802 CPU in software! Displays internal status as it simulates user programs for debugging, improvement, or learning 1802 operation. A must for every programmer! Intervene to alter simulation dynamically, by changing parameters, overriding program branch decisions, etc. Memory protected! CSTP cannot be bombed by the worst test program or user key-in errors! Needs 2.5K RAM, plus user program area.

COMING SOON: Assembler; Text Editor; Text Formatter; I/O for 1802 PILOT!

TAILORED to your system! Write for details and low prices.

CUDDLY SOFTWARE
Dept. K43
157 CHARTER ROAD ✓228
WETHERSFIELD, CT. 06109

DISK DRIVE/CRT
★SALE★

Shugart SA801R for RS MOD. II ONLY

$485

Hazeltine 1000 (unused)................$499
Shugart SA400................................$279
Pertec FD200......$279—FD250......$359
MPI B51............$279—B52............$349
SA801R w/PS/Cab..........................$750
Dual Drives w/PS/Cab....................$1640

LIMITED QUANITITES

MTI 3304 W. MacArthur Blvd. Santa Ana, CA 92704
(714) 979-9923 ✓219

CONSULTANTS WANTED

Earn $200 to $400 per day and enjoy more independence working as a professional consultant. The computer industry's needs for qualified consultants are at an all-time high. A fast way to get listed in the NATIONAL REGISTER OF COMPUTER CONSULTANTS AND SOFTWARE HOUSES. The 1980 edition of the REGISTER will be published July 15. Write or phone (without any obligation) for Registration Form and Profile Card.

ESSEX PUBLISHING CO.
285 Bloomfield Avenue
Caldwell, N.J. 07006 ✓342
(201) 783-6940

```
2020 E1=Z3
2030 GOSUB 2430
2040 FOR I1=1 TO Z3
2050   I0=I0+1
2060   B2=INT(I1/B1)
2070   A=I1-B2*B1
2080   IF A>0 THEN B3=B2+1
2090   IF A=0 THEN A=B1
2100   B4=B3+3
2110   C$=S$(I1*U0-5,I1*U0-3)
2120   CONVERT C$ TO B
2130   C$=S$(I1*U0-2,I1*U0)
2140   CONVERT C$ TO R
2150   REM B IS BLOCK IDENTIFIER TAKEN FROM SORT STRING.  IT IS
2160   REM USED TO GET FULL RECORD INFORMATION FROM SOURCE FILE.
2170   REM R IS NOW USED TO POINT TO SOURCE RECORD IN BLOCK.
2180   REM ONCE THE SOURCE FILE BLOCK IS READ, THE APPROPRIATE
2190   REM IS PLUCKED FROM A$ AND PLACED INTO G$
2200   GET (1,E,A$,B)
2210   G$(A*D-(D-1),A*D)=A$(R*D-(D-1),R*D)
2220   REM I0 IS USED TO KEEP TRACK OF THE NUMBER OF RECORDS
2230   REM PLACED INTO G$.  ONCE THE COUNT I0 EQUALS THE BLOCKING
2240   REM FACTOR B1, A FULL BLOCK IS WRITTEN TO THE DISKETTE.
2250   IF I0<B1 THEN 2290
2260   PUT (2,E,G$,B4)
2270   G$=F$
2280   I0=0
2290   NEXT I1
2300 IF G$=F$ THEN 2320
2310 PUT (1,E,G$,B4)
2320 #"" : CURSOR 7,5 : #"Sort completed"
2330 FOR I=1 TO 500 : # : NEXT I
2340 CLOSE (1,E) : CLOSE (2,E)
2350 #"******** End of processing *********"
2360 END
2370 REM ERROR ON DISK ROUTINES
2380 #"Error code is ";E
2390 IF E>0 THEN #"ALL FILE HAVE BEEN CLOSED"
2400 IF E>0 THEN CLOSE (1,E) : CLOSE (2,E)
2410 STOP : RUN#0,"DB0001"
2420 END
2430 REM SORT ROUTINE BEGINS HERE -- BUBBLE SORT
2440 E1=Z3
2450 CURSOR 8,10 : #"SORTING "
2460 FOR A=1 TO E1-1
2470   FOR B0=A+1 TO E1
2480     X=A*U0-U0+1
2490     X9=B0*U0-U0+1
2500     IF O0=2 THEN 2530
2510     IF S$(X,X+U-1)<S$(X9,X9+U-1) THEN 2570
2520     GOTO 2540
2530     IF S$(X,X+U-1)>S$(X9,X9+U-1) THEN 2570
2540     G$=S$(X,X+U0-1)
2550     S$(X,X+U0-1)=S$(X9,X9+U0-1)
2560     S$(X9,X9+U0-1)=G$
2570     NEXT B0
2580   NEXT A
2590 B=B0
2600 G$=F$
2610 #""
2620 RETURN
2630 REM SORT ROUTINE
2640 E1=K0
2650 FOR A=1 TO E1-1
2660   FOR B0=A+1 TO E1
2670     IF O(A)<O(B0) THEN 2710
2680     T=O(A) : T1=V(A)
2690     O(A)=O(B0) : V(A)=V(B0)
2700     O(B0)=T : V(B0)=T1
2710     NEXT B0
2720   NEXT A
2730 RETURN
2740 REM SCREEN PRINT ROUTINE
2750 #"" : REM CLEAR SCREEN AND HOME CURSOR WITH #"CNTL L"
2760 D1=0
2770 J1=1
2780 FOR I1=1 TO Z
2790   CURSOR R(I1),C(I1)
2800   K1=J1+L(I1)-1
2810   #E$(J1,K1);
2820   #" ";
2830   J1=K1+1
2840   D1=D1+D(I1)
2850   FOR M1=1 TO D(I1)
2860     #"x";
2870     NEXT M1
2880   NEXT I1
2890 CURSOR 15,0
2900 #"                                                       ";
2910 CURSOR 15,0
2920 RETURN
```

A Line Editor For Benton Harbor BASIC

This program gives any Heath tape-based interpreter line-editing capabilities.

James F. Teixeira
62 Churchill St.
Sudbury, MA 01776

I have been using a Heathkit H8 system for nearly a year and am generally impressed with the quality of their design, both hardware and software. Some observers have stated that Heath's BASIC is inferior to others available—Microsoft BASIC is often cited as the top choice—but in many ways Heath outperforms its competitors. Features such as command completion, error detection during input, versatile tape routines and, of course, excellent documentation are among its better attributes.

However, Heath's BASIC does lack a line-editing capability. Once a line of text has been entered via a carriage return, it can be modified only by manually reentering the entire line with the change. To make matters worse, based on the laws of probability, the longer lines have more chance for error.

The program described here adds a useful line-editing capability to any of Heath's tape-based BASIC interpreters. The technique used is general enough to be applied to other BASIC programs if certain key memory locations can be determined.

Programmer interaction with EDIT is simple yet versatile. EDIT operates on the last line of text sent to the terminal. Performing the LIST command in BASIC on the desired line number will set this up. Then pressing the ESC key will activate the editor, which will wait for you to enter a search character. The line of text being edited will be sent to BASIC up to and including the search character, and EDIT will enter the modify mode. You can now talk directly to BASIC, just as if you had manually typed in the text up to the cursor position. Text may be deleted to the left of the cursor and added to the right.

When the modification is complete, you can choose to either perform another ESC-search sequence to edit further down the line or type a carriage return to enter the rest of the line from the search character onward. Also, if the stopping place is not the first occurrence of the search character, multiple ESC-search inputs can be used to position the cursor.

To illustrate this process, the following line:

```
*100 PRINT "THIS IS A LINE EDITOR"
```

will be changed to:

```
*100 PRINT "THIS IS THE LINE EDITOR":
GOTO 10
```

After first performing the LIST 100 command to get line 100 as the last line displayed, an ESC followed by an A is entered, and the following will result:

```
*100 PRINT "THIS IS A
```

The cursor will stop after the A. Entering a back space and THE will give:

```
*100 PRINT "THIS IS THE
```

An ESC followed by a quotation mark (") will move the cursor to the end of the line:

```
*100 PRINT "THIS IS THE LINE EDITOR"
```

Now, entering : GOTO 10, followed by a carriage return, will complete the edit:

```
*100 PRINT "THIS IS THE LINE EDITOR":
GOTO 10
```

Note that a carriage-return entry at any point in the process will always send the rest of the line to BASIC.

This editor also allows the line number to be changed using the same editing procedure described in the above example. The search character would be the last digit of the line number. A series of back spaces followed by the new line number and a carriage return will enter the text with the new number. This will not change the original line in memory, which can either be deleted or left in if a duplicate line of text is needed.

How EDIT Works

To understand how EDIT works first requires an understanding of Heath's Console Driver routine. This is the program that is called by all of

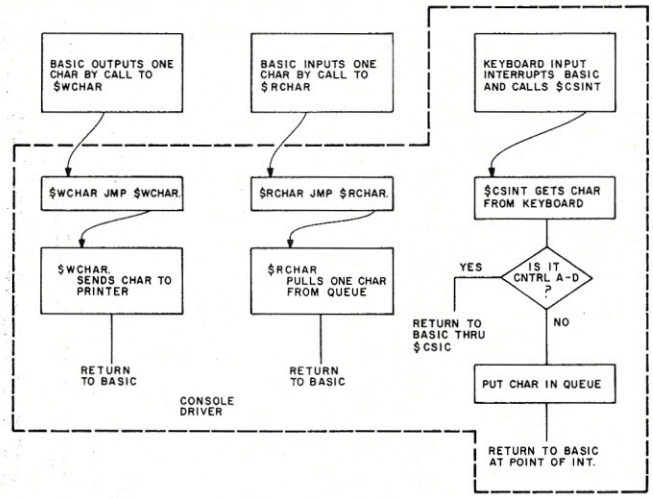


Fig. 1. Normal BASIC/Console Driver interaction.

Heath's software for sending and receiving characters to and from the video terminal or TTY-type I/O device. A source listing of it is conveniently provided in Heath's documentation.

Fig. 1 shows how the BASIC interpreter links with the Driver. Whenever BASIC outputs a character, it calls $WCHAR, which contains a JMP instruction to $WCHAR., the routine that actually sends the character to the I/O device. A similar process takes place for inputting a character: BASIC calls $RCHAR, which jumps to $RCHAR., the routine that pulls characters from a queue maintained by an interrupt-driven input program called $SCINT.

$SCINT processes console keyboard inputs and stuffs them in a first-in, first-out queue. It also monitors for special control-character inputs and, upon receipt of a CONTROL-A through CONTROL-D, will jump back to BASIC through the $CSIC vector. These control characters will not be placed in the queue.

EDIT links with BASIC through the Console Driver as shown in Fig. 2. The jump addresses at $WCHAR and $RCHAR are changed to EDIT's OUTCHR and INCHR entry points so that EDIT gets a chance to look at the text sent from BASIC and modify the text sent to BASIC.

EDIT also calls $WCHAR. and $RCHAR. as required. The $CSINT routine is also modified to make it jump to EDIT's CONC routine where a CONTROL-C check is made. If one is typed, EDIT will terminate and CONC will return through the $CSIC vector so that BASIC will also process the control character.

EDIT's OUTCHR and INCHR routines form the two main modules of the EDIT program. OUTCHR gets involved first. Its primary function is to save, on a line-by-line basis, the characters being output by BASIC. A 132-character buffer in this routine always contains the last full line of text that was output. The buffer starts at BUFST, and the present character position in it is pointed to by PNTR. PNTR is set to zero at each carriage return output by BASIC. It will not start moving from zero until a decimal digit is output.

Therefore, if you issue the command LIST 100 into BASIC, the buffer will end up with the line number (100) plus the line of text for that line number. The characters sent by BASIC would include carriage returns, line feeds and the prompt, all of which would be rejected since they occur when the pointer is at zero and is waiting for a decimal digit.

EDIT's INCHR routine is constantly monitoring what you are keying into BASIC. It is always in one of the following three modes.

1. Normal BASIC. The editor has not been called, and inputs are sent to BASIC directly.

2. Search Mode. The editor has been activated by an ESC character. The next input is a search character, and the editor will transmit the text in the buffer up to and including it.

3. Modify Mode. This mode automatically follows completion of a successful search and places you in direct communication with BASIC for text modification or addition. This mode can be exited by either another ESC-search sequence, a carriage return or a CONTROL-C.

Heath's BASIC performs command completion and dynamic error detection during input for illegal character combinations. As an example of command completion, if you type PR, BASIC will output the entire word PRINT to the I/O device. This complicates the edit program since the line buffer will contain the full word PRINT, but during the search mode only the PR part can be sent to BASIC.

EDIT handles this by having OUTCHR sense when the editor is in the search mode and advancing the character pointer for the line buffer whenever a character is output by BASIC. INCHR will not see the INT because after it scans the R and sends it to BASIC, OUTCHR will advance the pointer to the next character after the T.

This reflects the way you choose search characters for commands. If the command PRINT is to be changed, the search character must be the R (the last letter typed for the command). The cursor will advance to the end of the command, and a single back space can be used to delete the entire word.

Dynamic error correction in Heath's BASIC prevents your entering certain detectable invalid character combinations, such as a variable with two alphabetic characters. This presents no problem in the modify mode since you are talking directly to BASIC and must fix the error before BASIC will allow you to continue.

However, you may do something that appears valid at the time of entry but causes characters in a following search mode or line entry (due to a carriage return) to be invalid. OUTCHR performs a special test for the condition and will stop the

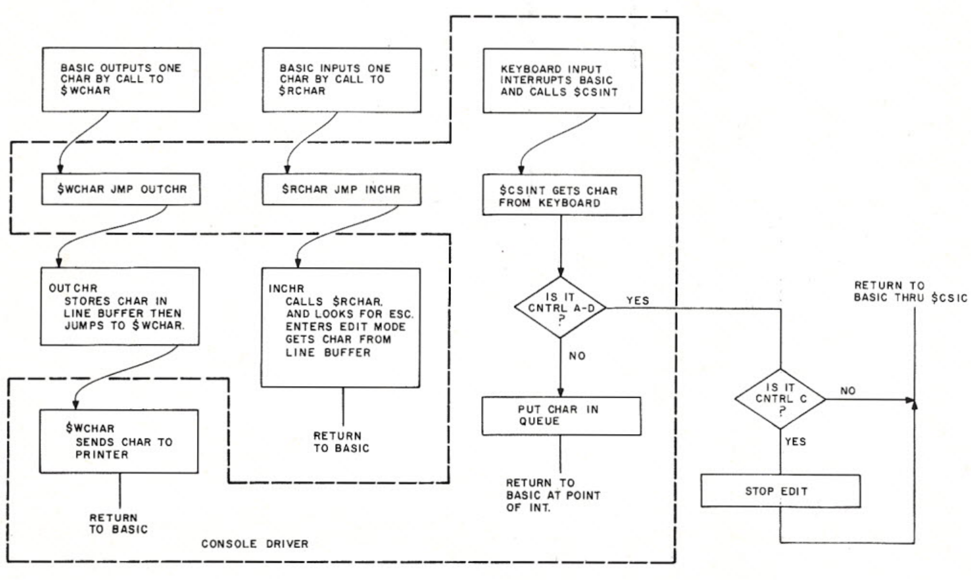


Fig. 2. BASIC/Console Driver/EDIT interaction.

search or line-completion entry at the point of error and will return EDIT back to the modify mode. If it is too late to correct the error, it's best to hit CONTROL-C to terminate EDIT and try again from the LIST step.

The program listing occupies 216 bytes plus an additional 132 bytes for the line buffer. The program consists of the modules INIT, INCHR, OUTCHR, CONC and two subroutines, INCPTR and BUFADD. RCHAR and WCHAR have been described above. INIT links EDIT with BASIC during the first cold start into BASIC by modifying the jump addresses in the Console Driver. CONC is a special routine used to process CONTROL characters and terminate EDIT on a CONTROL-C input. The two subroutines manipulate the buffer pointer and form addresses in the buffer for the next character to go in or come out.

Locating the Editor

The easiest place to locate EDIT is in the topmost part of RAM since BASIC can be configured from the distribution tape to avoid using this area. The program listing shows the program ORGed at the top of a 24K RAM system. This can be changed to any RAM limit by specifying MEMTOP as the address of the highest RAM location available (177377 for 24K, 137377 for 16K, etc.). EDIT is set up so that the line buffer occupies the top 132 RAM locations. This allows you to execute a RESET without harming the editor. The stack pointer, which is set to the top of RAM by RESET, will fall into the buffer area and not the program area.

Loading BASIC/EDIT

The first step in getting BASIC/EDIT into your system is easy. Starting with the Heath distribution tape for BASIC, set up your configuration parameters and include setting HIGH MEMORY to 350 (decimal) locations below the default limit displayed (for example, HIGH MEMORY = 32767/32417 for a 24K system). Execute the SAVE command to generate a configured BASIC tape.

The next step is more difficult

Program listing.

```
                         *           ******************
                         *           *                *
                         *           *   BASIC/EDIT   *
                         *           *                *
                         *           ******************
                         *
                         *        A LINE EDITING PROGRAM FOR HEATHKIT TAPE BASIC
                         *
                         *        3/25/79
                         *
                         *        JAMES F. TEIXEIRA
                         *        62 CHURCHILL ST.
                         *        SUDBURY MA. 01776
                         *
                         *
                         * SET MEMTOP AS THE HIGHEST RAM LOCATION IN YOUR SYSTEM.
                         * SET HIGH MEMORY IN BASIC TO BE 350 (DECIMAL) LOCATIONS
                         * BELOW HIGHEST RAM.
                         *
177.377                  MEMTOP   EQU      177377A   * TOP OF 24K RAM SYSTEM
                         *
                         * FOR VERSION 10.05.00 OF EXT. BASIC, USE THE VALUES IN BRACKETS
                         *
040.332                  $WCHAR.  EQU      40332A    * BASIC'S WRITE ONE CHAR ROUTINE [40347A]
040.255                  $RCHAR.  EQU      40255A    * BASIC'S READ ONE CHAR ROUTINE [40302A]
040.252                  $CSLCTL  EQU      40252A    * BASIC'S CONTROL REGISTER FLAGS [40214A]
040.250                  $CSIC    EQU      40250A    * RETURN ADD FOR CNTRL-A THRU -E [40212A]
                         *
                         *
176.207                           ORG      MEMTOP-376
                         *
                         *
                         * INIT SETS UP EDIT ENTRY VECTORS IN THE CONSOLE DRIVER.
                         * INIT IS USED ONLY FOR THE FIRST COLD START INTO BASIC.
                         * IT MAY BE DESTROYED BY THE USER PROGRAM.
                         *
176.207  041 004 177     INIT     LXI      H,INCHR   * SET UP INCHR JUMP VECTOR
176.212  042 145 040              SHLD     40145A    * [40107A]
176.215  041 241 176              LXI      H,OUTCHR  * SAME FOR OUTCHR
176.220  042 150 040              SHLD     40150A    * [40112A]
176.223  041 146 177              LXI      H,CONC    * SAME FOR CNTRL-C TEST
176.226  042 141 041              SHLD     41141A    * [41315A]
176.231  076 303                  MVI      A,303A    * CNTRL-C TO BE JUMP INSTRUCTION
176.233  062 140 041              STA      41140A    * [41314A]
176.236  303 100 040              JMP      40100A    * ENTER BASIC
                         *
                         * OUTCHR SAVES THE LAST LINE OF TEXT DISPLAYED IN BUFST INDEXED
                         * BY PNTR. IT ALSO ADVANCES PNTR DURING THE SEARCH MODE IN EDIT
                         * TO ALLOW SEARCHING THRU COMMAND COMPLETION. INVALID ENTRY
                         * DURING THE SEARCH MODE CAUSES EDIT TO STOP AT THE INVALID CHAR.
                         *
176.241  345             OUTCHR   PUSH     H         * SAVE REGISTERS
176.242  305                      PUSH     B
176.243  325                      PUSH     D
176.244  365                      PUSH     PSW
176.245  127                      MOV      D,A       * SAVE OUTPUT CHAR IN D
176.246  315 260 176              CALL     OUTCHR1   * SAVE LINE ROUTINE
176.251  361                      POP      PSW       * RESTORE REGISTERS
176.252  321                      POP      D
176.253  301                      POP      B
176.254  341                      POP      H
176.255  303 332 040              JMP      $WCHAR.   * RETURN TO BASIC
                         *
                         *
176.260  072 166 177     OUTCHR1  LDA      EDFLG     * SEE IF IN EDIT MODE
176.263  247                      ANA      A         * SET FLAGS
176.264  312 315 176              JZ       OUTCHR2   * NO, SAVE OUTPUT IN BUFFER
176.267  370                      RM                 * IN MODIFY MODE, RETURN
176.270  172                      MOV      A,D       * MUST BE IN SEARCH MODE
176.271  376 007                  CPI      7         * SEE IF 'BEL' WAS OUTPUTTED
176.273  302 353 176              JNZ      OUTCHR4   * NO, SEE IF BACKSPACE
176.276  315 371 176              CALL     BUFADD    * SEE IF 'BEL' WAS SENT
176.301  176                      MOV      A,M       * GET LAST CHAR IN BUFFER
176.302  376 007                  CPI      7         * WAS IT 'BEL'
176.304  312 356 176              JZ       INCPTR    * YES, TREAT AS NORMAL
176.307  076 200                  MVI      A,200Q    * ERROR DURING SEARCH, STOP IT
176.311  062 166 177              STA      EDFLG
176.314  311                      RET
176.315  072 167 177     OUTCHR2  LDA      PNTR      * TEST PNTR FOR ZERO
176.320  247                      ANA      A         * SET FLAGS
176.321  302 333 176              JNZ      OUTCHR3   * PNTR NOT ZERO, PROCESS NORMAL
176.324  172                      MOV      A,D       * PNTR AT ZERO, NEED DIGIT TO START
176.325  376 060                  CPI      60Q       * 60Q = DIG 0 IN ASCII
176.327  330                      RC                 * OUT OF DIGIT RANGE
176.330  376 072                  CPI      72Q       * 71Q = DIG 9 IN ASCII
176.332  320                      RNC                * OUT OF DIGIT RANGE
176.333  315 371 176     OUTCHR3  CALL     BUFADD    * GET CHAR ADDRESS IN BUFFER
176.336  162                      MOV      M,D       * STORE IT IN BUFFER
176.337  315 356 176              CALL     INCPTR    * MOVE POINTER TO NEXT POSITION
176.342  172                      MOV      A,D       * GET CHAR
176.343  376 015                  CPI      15Q       * WAS IT CR?
176.345  300                      RNZ                * NO, RETURN
176.346  257                      XRA      A         * YES, SET PNTR TO 0 FOR NEXT LINE
176.347  062 167 177              STA      PNTR
176.352  311                      RET
176.353  376 010         OUTCHR4  CPI      10Q       * IS IT BACKSPACE?
176.355  310                      RZ                 * YES, RETURN, ELSE INC. PNTR
                         *
                         *
```

```
                              * THE FOLLOWING ROUTINE INCREMENTS PNTR
                              *
176.356  072 167 177  INCPTR  LDA     PNTR      * GET POINTER IN A
176.361  074                  INR     A         * INCREMENT IT
176.362  376 204              CPI     204Q      * MAX. 132 CHAR ALLOWED
176.364  310                  RZ                * TRIED TO EXCEED IT
176.365  062 167 177  INCPT1  STA     PNTR      * PUT IT BACK
176.370  311                  RET
                              *
                              *
                              * THE FOLLOWING ROUTINE FORMS THE ADD. OF THE CURRENT CHAR IN
                              * THE BUFFER BY ADDING PNTR TO BUFST
                              *
176.371  072 167 177  BUFADD  LDA     PNTR      * GET POINTER IN A
176.374  117                  MOV     C,A       * FORM POINTER IN B,C
176.375  006 000              MVI     B,0
176.377  041 170 177          LXI     H,BUFST   * START OF BUFFER
177.002  011                  DAD     B         * ADD POINTER TO BUFFER START
177.003  311                  RET
                              *
                              *
                              * THE FOLLOWING ROUTINE LOOKS FOR AN 'ESC' CHAR TO START THE EDIT
                              * FUNCTION. FOLLOWING AN 'ESC' IT EXPECTS A SEARCH CHAR. AND WILL
                              * OUTPUT THE LINE IN THE BUFFER UP TO THIS CHAR. THE USER CAN THEN
                              * ADD OR DELETE TEXT OR ADVANCE THE LINE OF TEXT VIA ANOTHER 'ESC'-
                              * SEARCH CHAR. SEQUENCE. A 'CR' WILL CAUSE THE REST OF THE LINE TO
                              * BE SENT TO BASIC.
                              *
177.004  072 166 177  INCHR   LDA     EDFLG     * SEE WHAT PRESENT MODE IS,
                      *                           0=NORMAL, 200Q=MODIFY,
                      *                           177Q=SEARCH END PENDING
                      *                           ANYTHING ELSE=SEARCH MODE
177.007  247                  ANA     A         * SET FLAGS
177.010  302 033 177          JNZ     INCHR1    * IN EDIT MODE
177.013  315 255 040          CALL    $RCHAR.   * NOT IN EDIT, GET INPUT
177.016  376 033              CPI     33Q       * IS IT ESC?
177.020  300                  RNZ               * NO, RETURN
177.021  072 167 177          LDA     PNTR      * YES, POINTER MUST BE AT ZERO FOR EDIT
177.024  247                  ANA     A
177.025  312 075 177          JZ      INCHR3    * OK, IT'S AT ZERO
177.030  076 033              MVI     A,33Q     * NOT AT ZERO, RETURN WITH ESC IN A
177.032  311                  RET
177.033  376 177      INCHR1  CPI     177Q      * IS IT SEARCH END PENDING
177.035  332 103 177          JC      INCHR5    * NO, IN SEARCH MODE
177.040  302 057 177          JNZ     INCHR2    * NO, IN MODIFY MODE
177.043  072 167 177          LDA     PNTR      * SEARCH MODE WAS PENDING
177.046  075                  DCR     A         * DECREMENT PNTR
177.047  062 167 177          STA     PNTR
177.052  076 200              MVI     A,200Q    * AND SET TO MODIFY MODE
177.054  062 166 177          STA     EDFLG
177.057  315 255 040  INCHR2  CALL    $RCHAR.   * IN MODIFY MODE, GET CHAR
177.062  376 033              CPI     33Q       * IS IT ESC?
177.064  312 075 177          JZ      INCHR3    * YES, START NEW SEARCH
177.067  376 015              CPI     15Q       * IS IT CR?
177.071  312 100 177          JZ      INCHR4    * YES, EDIT COMPLETE
177.074  311                  RET               * TREAT AS MODIFY INPUT
177.075  315 255 040  INCHR3  CALL    $RCHAR.   * GET INPUT, ITS A SEARCH CHAR
177.100  062 166 177  INCHR4  STA     EDFLG     * SAVE IT
177.103  345          INCHR5  PUSH    H
177.104  305                  PUSH    B
177.105  315 371 176          CALL    BUFADD    * FORM BUFFER ADDRESS
177.110  176                  MOV     A,M       * GET CHAR FROM BUFFER
177.111  117                  MOV     C,A       * SAVE IT IN C
177.112  376 015              CPI     15Q       * WAS IT A CR?
177.114  302 123 177          JNZ     INCHR6    * NOT CR
177.117  257                  XRA     A         * YES, CLEAR EDIT MODE
177.120  303 137 177          JMP     INCHR7
177.123  072 166 177  INCHR6  LDA     EDFLG     * GET SEARCH CHAR
177.126  271                  CMP     C         * IS IT SAME AS IN BUFFER
177.127  302 142 177          JNZ     INCHR8    * NO
177.132  315 356 176          CALL    INCPTR    * CHAR FOUND, INC. POINTER
177.135  076 177              MVI     A,177Q    * FLAG END OF SEARCH PENDING
177.137  062 166 177  INCHR7  STA     EDFLG
177.142  171          INCHR8  MOV     A,C       * GET CHAR
177.143  301                  POP     B
177.144  341                  POP     H
177.145  311                  RET
                              *
                              *
                              * TEST FOR CONTROL C, MEANS LEAVE EDIT MODE WITHOUT ALTERING LINE
                              *
177.146  072 252 040  CONC    LDA     $CSLCTL   * CONTROL CHAR FLAG BITS
177.151  346 040              ANI     40Q       * CONTROL C FLAG
177.153  312 162 177          JZ      CONC1     * NOT CONTROL C
177.156  257                  XRA     A         * CLEAR EDIT MODE
177.157  062 166 177          STA     EDFLG
177.162  052 250 040  CONC1   LHLD    $CSIC     * JUMP TO NORMAL CONTROL PROCESSING
177.165  351                  PCHL
                              *
                              *
                              * STORAGE LOCATIONS
                              *
                              *
177.166  000          EDFLG   DB      0         * FLAG REGISTER FOR PRESENT MODE
177.167  000          PNTR    DB      0         * POINTER FOR LINE BUFFER
177.170  000          BUFST   DB      0         * START OF LINE BUFFER, EXTENDS TO MEMTOP
                              *
                              *
177.171                       END     INIT      * SETS PROG. TO START AT INIT
```

since you must now generate a tape of the EDIT program shown in the program listing. It can be hand keyed in from the listing directly for a 24K RAM system or hand modified or reassembled to fit your memory size. Once in memory, a tape can be generated using the DUMP button on the H8. Set the Program Counter to the INIT address before dumping the tape so that EDIT will automatically start there.

These two programs form BASIC/EDIT. They can both be on the same tape with BASIC first, followed immediately by EDIT. To use them, first load the BASIC part into your system. Don't press GO. Then load in the EDIT portion. Now press GO, and the INIT routine will set up the Console Driver to link with EDIT and will then jump to the normal BASIC cold-start address.

BASIC/EDIT can now be used in exactly the same way as normal BASIC except it will respond to the edit commands. Your programs can be loaded, dumped, run or edited. Note that the configured BASIC tape can be used by itself by pressing GO immediately after loading it. The editor part only has to be loaded if you foresee a need to do line editing.

Conclusion

EDIT can be entered only if the first ESC is done when the cursor is at the beginning of the line (at the prompt character). This prevents accidental entry into EDIT in the middle of a manual entry. Don't forget that a CONTROL-C can be used at any time in the edit process to recover from a botched-up edit sequence. Finally, multiple ESC-search entries can be made to position the cursor at the desired place in the line if you want to stop at a point that is not at the first occurrence of the character.

I am sure you will enjoy using BASIC/EDIT and will wonder why Heathkit didn't provide it as part of their BASIC. I find it so useful that I am presently modifying it to work with Heath's Text Editor (TED-8), which does not have this type of character-oriented editing capability. ■

NEW! TPM* for TRS-80 Model II
NEW! System/6 Package
Computer Design Labs

Z80* Disk Software

We have acquired the rights to all TDL software (& hardware). TDL software has long had the reputation of being the best in the industry. Computer Design Labs will continue to maintain, evolve and add to this superior line of quality software.
— Carl Galletti and Roger Amidon, owners.

Software with Manual/Manual Alone

All of the software below is available on any of the following media for operation with a Z80 CPU using the CP/M* or similar type disk operating system (such as our own TPM*).

for TRS-80* CP/M (Model I or II)
for 8" CP/M (soft sectored single density)
for 5¼" CP/M (soft sectored single density)
for 5¼" North Star CP/M (single density)
for 5¼" North Star CP/M (double density)

BASIC I
A powerful and fast Z80 Basic interpreter with EDIT, RENUMBER, TRACE, PRINT USING, assembly language subroutine CALL, LOADGO for "chaining", COPY to move text, EXCHANGE, KILL, LINE INPUT, error intercept, sequential file handling in both ASCII and binary formats, and much, much more. It runs in a little over 12 K. An excellent choice for games since the precision was limited to 7 digits in order to make it one of the fastest around. $49.95/$15.

BASIC II
Basic I but with 12 digit precision to make its power available to the business world with only a slight sacrifice in speed. Still runs faster than most other Basics (even those with much less precision). $99.95/$15.

NEW MANUAL

BUSINESS BASIC
The most powerful Basic for business applications. It adds to Basic II with random or sequential disk files in either fixed or variable record lengths, simultaneous access to multiple disk files, PRIVACY command to prohibit user access to source code, global editing, added math functions, and disk file maintenance capability without leaving Basic (list, rename, or delete). $179.95/$25.

ZEDIT
A character oriented text editor with 26 commands and "macro" capability for stringing multiple commands together. Included are a complete array of character move, add, delete, and display function. $49.95./$15.

ZTEL
Z80 Text Editing Language - Not just a text editor. Actually a language which allows you to edit text and also write, save, and recall programs which manipulate text. Commands include conditional branching, subroutine calls, iteration, block move, expression evaluation, and much more. Contains 36 value registers and 10 text registers. Be creative! Manipulate text with commands you write using Ztel. $79.95/$25.

TOP
A Z80 Text Output Processor which will do text formatting for manuals, documents, and other word processing jobs. Works with any text editor. Does justification, page numbering and headings, spacing, centering, and much more! $79.95/$25.

MACRO I
A macro assembler which will generate relocateable or absolute code for the 8080 or Z80 using standard Intel mnemonics plus TDL/Z80 extensions. Functions include 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical operations, local and global symbols, chaining files, linking capability with optional linker, and recursive/reiterative macros. This assembler is so powerful you'll think it is doing all the work for you. It actually makes assembly language programming much less of an effort and more creative. $79.95/$20.

MACRO II
Expands upon Macro I's linking capability (which is useful but somewhat limited) thereby being able to take full advantage of the optional Linker. Also a time and date function has been added and the listing capability improved. $99.95/$25.

LINKER
How many times have you written the same subroutine in each new program? Top notch professional programmers compile a library of these subroutines and use a Linker to tie them together at assembly time. Development time is thus drastically reduced and becomes comparable to writing in a high level language but with all the speed of assembly language. So, get the new CDL Linker and start writing programs in a fraction of the time it took before. Linker is compatible with Macro I & II as well as TDL/Xitan assemblers version 2.0 or later. $79.95/$20.

DEBUG I
Many programmers give up on writing in assembly language even though they know their programs would be faster and more powerful. To them assembly language seems difficult to understand and follow, as well as being a nightmare to debug. Well, not with proper tools like Debug I. With Debug I you can easily follow the flow of any Z80 or 8080 program. Trace the program one step at a time or 10 steps or whatever you like. At each step you will be able to see the instruction executed and what it did. If desired, modifications can then be made before continuing. It's all under your control. You can even skip displaying a subroutine call and up to seven breakpoints can be set during execution. Use of Debug I can pay for itself many times over by saving you valuable debugging time. $79.95/$20.

DEBUG II
This is an expanded debugger which has all of the features of Debug I plus many more. You can "trap" (i.e. trace a program until a set of register, flag, and/or memory conditions occur). Also, instructions may be entered and executed immediately. This makes it easy to learn new instructions by examining registers/memory before and after. And a RADIX function allows changing between ASCII, binary, decimal, hex, octal, signed decimal, or split octal. All these features and more add up to give you a very powerful development tool. Both Debug I and II must run on a Z80 but will debug both Z80 and 8080 code. $99.95/$20.

ZAPPLE
A Z80 executive and debug monitor. Capable of search, ASCII put and display, read and write to I/O ports, hex math, breakpoint, execute, move, fill, display, read and write in Intel or binary format tape, and more! on disk

APPLE
8080 version of Zapple

NEW

NEW! TPM now available for TRS-80 Model II!

TPM*
A NEW Z80 disk operation system! This is not CP/M*. It's better! You can still run any program which runs with CP/M* but unlike CP/M* this operating system was written specifically for the Z80* and takes full advantage of its extra powerful instruction set. In other words its not warmed over 8080 code! Available for TRS-80* (Model I or II). Tarbell, Xitan DDDC, SD Sales "VERSA-FLOPPY", North Star (SD&DD), and Digital (Micro) Systems. $79.95/$25.

SYSTEM MONITOR BOARD (SMB II)
A complete I/O board for S-100 systems. 2 serial ports, 2 parallel ports, 1200/2400 baud cassette tape interface, sockets for 2K of RAM, 3-2708/2716 EPROM's or ROM, jump on reset circuitry. Bare board $49.95/$20.

ROM FOR SMB II
2KX8 masked ROM of Zapple monitor. Includes source listing $34.95/$15.

PAYROLL (source code only)
The Osborne package. Requires C Basic 2.
5" disks $124.95 (manual not included)
8" disks $ 99.95 (manual not included)
Manual $20.00

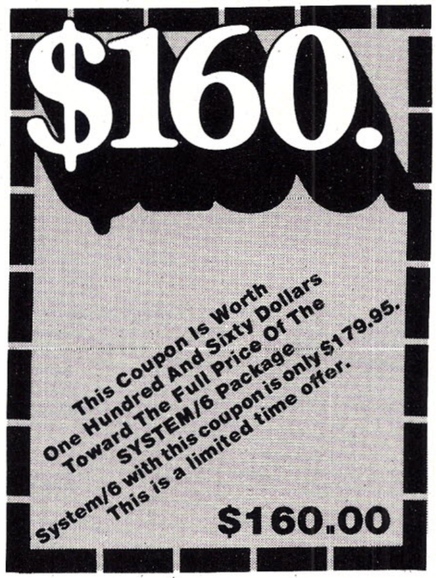
ACCOUNTS PAYABLE/RECEIVABLE (source code only)
By Osborne, Requires C Basic 2
5" disks $124.95 (manual not included)
8" $99.95 (manual not included)
Manual $20.00

GENERAL LEDGER (source code only)
By Osborne. Requires C Basic 2
5" disks $99.95 (manual not included)
8" disks $99.95 (manual not included)
Manual $20.00

C BASIC 2
Required for Osborne software. $99.95/$20.

SYSTEM/6
NEW

TPM with utilities, Basic I interpreter, Basic E compiler, Macro I assembler, Debug I debugger, and ZEDIT text editor.
Above purchased separately costs $339.75
Special introductory offer: Only $179.75 with coupon!!



ORDERING INFORMATION
Visa, Master Charge and C.O.D. O.K. To order call or write with the following information.

1. Name of Product (e.g. Macro I)
2. Media (e.g. 8" CP/M)
3. Price and method of payment (e.g. C.O.D.) include credit card info. if applicable.
4. Name, Address and Phone number.
5. For TPM orders only: Indicate if for TRS 80, Tarbell, Xitan DDDC, SD Sales (5¼" or 8"), ICOM (5¼" or 8"), North Star (single or double density) or Digital (Micro) Systems.
6. N.J. residents add 5% sales tax.

Manual cost applicable against price of subsequent software purchase in any item except for the Osborne software.

For information and tech queries call
609-599-2146

For phone orders ONLY call toll free
1-800-327-9191
Ext. 676
(Except Florida)

OEMS
Many CDL products are available for licensing to OEMs. Write to Carl Galletti with your requirements.

* Z80 is a trademark of Zilog
* TRS-80 is a trademark for Radio Shack
* TPM is a trademark of Computer Design Labs. It is not CP/M*
* CP/M is a trademark of Digital Research
Prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.

COMPUTER DESIGN LABS
342 Columbus Avenue
Trenton, N.J. 08629

✓18

16K MEMORY KITS
TMS—4116 250 N.S.

FOR: APPLE, TRS-80, HEATH EXPANDORAM I & II AND MANY MORE

12 MONTH GUARANTEE

$59.95 PER SET OF 8 CHIPS

ADD $2.00 SHIPPING & HANDLING PER ORDER
IF C.O.D. ADD $1.00.
PAYMENT: MASTER CHARGE, VISA CASH, MONEY ORDER, U.P.S., C.O.D., PERSONAL CHECKS REQUIRE 2-3 WKS. TO CLEAR BANK.

Computer Services ✓80

30 Hwy. 321, N.W.
P. O. Box 2292
Hickory, N. C. 28601
294-1616

PHONE HOURS:

6 P.M. — 9 P.M. EST

APPLE II PLUS WITH 48K RAM	$1340.
TEXAS INSTRUMENT 99/4 COMPUTER	$ 989.
TI 810 PRINTER	$1590.
CENTRONIC PRINTERS:	
730-1 PARALLEL PRINTER	$ 749.
737-1 PARALLEL INTERFACE	$ 879.
SAVE ON ALL OTHER MODELS	
SPINWRITERS FROM NEC	
5510 R/O SERIAL INTERFACE	$2690.
5520 KSR SERIAL WITH KEYBOARD	$2890.
5530 PARALLEL INTERFACE	$2690.
PAPER TIGER 440	$ 929.
440G	$ 990.
BASE-2 PRINTERS	
800 S.T.	$ 559.
800 M.S.T.	$ 599.
COMPRINT 912 APPLE, TRS-80, PET	$ 559.
912 SERIAL	$ 599.
KIM-1 W/MANUALS	$ 155.
SYM-1 W/MANUALS	$ 229.
COMMODORE BUSINESS MACHINES:	
CBM 8016	$1329.
8032	$1595.
PET 2001-8K COMPUTER	$ 695.
PET 2001-16K	$ 895.
PET 2001-32K	$1090.
PET 2022 TRAC. FEED PRINTER	$ 699.
PET 2023 FRIC. FEED PRINTER	$ 679.
PET 2040 DUAL FLOPPY DISK DRIVE	$1090.
PET 8050	$1499.
ATARI 800	$ 889.
400	$ 495.
INTERTEC SUPERBRAIN (32K)	$2595.
NORTH STAR COMPUTERS	
HRZ-2-32K-D-ASM	$2445.
HRZ-2-32K-Q-ASM	$2849.
DISPLAY TERMINALS:	
INTERTUBE II	$ 775.
HAZELTINE 1410	$ 785.
HAZELTINE 1420	$ 949.
SAVE ON COMPLETE HAZELTINE LINE	

IMMEDIATE DELIVERY FROM STOCK PRICES SUBJECT TO CHANGE

MULTI-BUSINESS COMPUTER SYSTEMS
28 MARLBOROUGH STREET ✓81
PORTLAND, CONN. 06480
M/F 9-6 SAT 9-3 (203) 342-2747

ALARM CLOCK KITS:
4 Digit .5"

Here it is! The first of several quality kits we have been asked for: Here is what you get — unbelievable as it may sound...

- 1 National - 5375AA Clock Chip
- 1 Bowmar Clock Stick Readout (L.E.D.) 4 digit - 1/2"
- 13 Transitors
- 2 Push Buttons for time set
- 2 Toggle Switches for alarm
- 1 Filter cap
- 4 1N4000 series diodes
- 1 1N4148
- 2 Disc caps
- 29 Resistors
- 1 Transducer (Speaker) for Alarm
- 1 LED Lamp for alarm indicator

NEW! **$9.99**

ORDER KIT CK-100AC

P.C. Board **$2.25**

Plug In Transformer **$1.50**

Case - $3.50

D.C. MODEL

Same as above except it includes 60 Hz timebase.

This Kit Includes:

- 1 National 5375AA Clock Chip
- 1 Bowmar Clock Stick Readout - (L.E.D.) 4 digit - 1/2"
- 12 Transistors
- 2 Push Buttons for time set
- 2 Disc caps
- 27 Resistors
- 1 MOV
- 1 60 Hz time base

NEW! **$12.75**

ORDER KIT CK-100DC

P.C. Board **$2.25**

Case - $3.50

CABLE TIES

MAKE YOUR PROJECTS "NEAT & TIDY". 4" CABLE TIES AT A FANTASTIC PRICE. GET THIS BARGAIN AND "TIE" IT DOWN. **$2.00 for 100 or better yet $15.00 for 1000**

FACTORY PRIME
BI - Polar LED 59¢ ea. or 10 for $5

LAB-BENCH VARIABLE POWER SUPPLY KIT

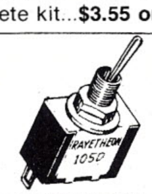
5 to 20 VDC at 1 AMP. Short circuit protected by current limit. Uses IC regulator and 10 AMP Power Darlington. Very good regulation and low ripple. Kit includes PC Board, all parts, large heatsink and shielded transformer. 50 MV. TYP. Regulation. **$15.99 KIT**

5-14 V.D.C. **BETTER BEEPER**

Two audio oscillators - a low frequency pulse oscillator - either or both audio frequencies can be shifted (warbled) or pulsed on and off. Constant tone capability. Any combination of pulses and warbles or tones. 1⅞"D X 1¾" ALARMS - TOYS - TESTING - ETC. Complete kit...**$3.55 or 3 for $9**

MICRO MINI TOGGLE SWITCHES

6 for $5 with hardware.



99¢ EACH

16K DYNAMIC RAM CHIP
WORKS IN TRS-80 OR APPLE II (300ns)
VERY LIMITED STOCK! "MAGAZINE SPECIAL" - **8/$59.95**

Digital Research: Parts
(OF TEXAS)
P.O. BOX 401247C GARLAND, TEXAS 75040 • (214) 271-2461

TERMS: Add 50¢ postage, we pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa, MasterCharge and American Express cards. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P&H. 90 Day Money Back Guarantee on all items. Write for our free catalog full of many useful bargains.

Calculating Interest Rates

The way interest rates wildly fluctuate, who can even remember what they are from day to day? You can—by using this North Star interest-rate-calculation program to keep track.

John A. Bryant
6648 N. Canandaigua Rd.
Holcomb, NY 14469

Most people can recall the principal amount and term of their loans, as well as the monthly payment, but few can remember the interest rate. Strangely, almost all amortization schedule programs require the input of the interest rate in order to run the program, although most programs will calculate the monthly payment if necessary. This means that if you don't know the interest rate you can't run the program.

Introduction

Recently I needed to determine the interest rate of a particular loan, so I decided to add a few lines to my amortization schedule program to calculate it when it is unknown. After all, you can simply rearrange the equation that calculates the monthly payment so that it is solved for the interest rate instead of the payment . . . or so I thought!

I listed my amortization schedule program and found the equation to be: M = P*(R*(1 + R)↑N)/(((1 + R)↑N) − 1), where

M = monthly payment
P = principal
R = monthly interest rate
N = number of months in loan

```
XXX INPUT "ANNUAL INTEREST RATE (ENTER 0 TO CALCULATE)? ";R
XYX IF R=0 THEN GOSUB YYY
```

Example 1.

Seventeen sheets of paper later, I concluded I couldn't solve the equation for R, so I started looking for a book or article in which someone smarter than I had done the work for me.

I found an article in the April 1979 issue of *Memory Pages*, the monthly magazine of the Rochester Area Microcomputer Society. The author, Reid Shay, stated that there is *no* equation to calculate interest rates!

Apparently the only way to calculate interest rates is by trial and error—just what computers do best. Mr. Shay had a short program in his article for finding the interest rate in precisely that manner, and I have adapted and expanded it into a form that should run with about any BASIC.

The only part of the program not strictly standard is line 360: A$ = A$(1,1). If your BASIC does not recognize the convention utilized in line 360, simply leave that line out. Those who have the LEFT$ function can rewrite the line as A$ = LEFT$(A$,1).

How the Program Works

My program in North Star BASIC uses the formula from my amortization schedule program and calculates a monthly payment using 50 percent as its first guess for the interest rate. It then compares the monthly payment so calculated to the monthly payment being made on the loan. If they are not the same, it will adjust the interest rate upward or downward, as the case may require, calculate again and then again compare the result to the monthly payment. This procedure is repeated until the two figures are the same (to the nearest penny), at which time the program prints the interest rate.

The program is not set up to calculate interest rates of over 100 percent. Where the interest rate equals or exceeds 100 percent, a message to that effect will be printed. (For those of you involved with loan sharks, simple changes in lines 190, 270 and 280 will permit calculation of interest rates over 100 percent.) Similarly, if the monthly payment is not sufficient to amortize the loan even at no interest, a message to that effect will be printed.

The program can be kept as a separate program as written and loaded and run as needed, or you may add it as a subroutine to your amortization schedule program so it will always be ready when you need it. Assume your amortization schedule program uses R as the variable for the interest rate. To add the program as a subroutine, delete lines 350 through 390, change line 340 to RETURN, modify the

```
                    INTEREST RATE CALCULATION

PRINCIPAL?  35000
MONTHLY PAYMENT?  275.35
HOW MANY YEARS (TYPE 0 TO ENTER MONTHS)?  30

INTEREST RATE IS 8.75%

WANT TO DO ANOTHER (Y/N)?  YES


PRINCIPAL?  1000
MONTHLY PAYMENT?  150.00
HOW MANY YEARS (TYPE 0 TO ENTER MONTHS)?  0
HOW MANY MONTHS?  6

PAYMENT SET TOO LOW FOR A 6 MONTH LOAN


PRINCIPAL?  1000
MONTHLY PAYMENT?  150.00
HOW MANY YEARS (TYPE 0 TO ENTER MONTHS)?  0
HOW MANY MONTHS?  7

INTEREST RATE IS 14.82%

WANT TO DO ANOTHER (Y/N)?  Y


PRINCIPAL?  1000
MONTHLY PAYMENT?  175.00
HOW MANY YEARS (TYPE 0 TO ENTER MONTHS)?  1

INTEREST RATE IS OVER 100%


PRINCIPAL?  32500
MONTHLY PAYMENT?  357.86
HOW MANY YEARS (TYPE 0 TO ENTER MONTHS)?  20

INTEREST RATE IS 12%

WANT TO DO ANOTHER (Y/N)?  NO THANKS
READY
```

Sample run.

input portion of the amortization schedule program as shown in Example 1 and alter the line numbers. To keep it simple you might alter the line numbers by adding 1000 or 2000 to each.

Unless you know the interest rate, you can't run most amortization programs. Thus, if you have an amortization program, you should add this short and simple program to your tape or disk for those times, certain to occur, when you don't know the interest rate of a particular loan. ■

```
10  REM***********************************************************
20  REM**                                                       **
30  REM** THIS PROGRAM CALCULATES THE INTEREST RATE WHERE THE   **
40  REM**  PRINCIPAL, MONTHLY PAYMENT, AND TERM OF LOAN ARE     **
50  REM**                     KNOWN.                            **
60  REM**                                                       **
70  REM***********************************************************
80  PRINT TAB(10); "INTEREST RATE CALCULATION"
90  PRINT
100 PRINT
110 INPUT "PRINCIPAL?  "; P
120 INPUT "MONTHLY PAYMENT?  "; M
130 INPUT "HOW MANY YEARS (TYPE 0 TO ENTER MONTHS)?  "; N
140 IF N>0 THEN N=N*12
150 IF N=0 THEN INPUT "HOW MANY MONTHS?  "; N
160 PRINT
170 IF N*M<P THEN PRINT "PAYMENT SET TOO LOW FOR A"; N; " MONTH LOAN"
180 IF N*M<P THEN 90
190 H=100
200 L=0
210 R=(H+L)/2
220 T=R/1200
230 M1=P*(T*(1+T)^N)/(((1+T)^N)-1)
240 IF M+.01>M1 THEN 300
250 IF M1<M THEN L=R
260 IF M1>M THEN H=R
270 IF R=100 THEN PRINT "INTEREST RATE IS OVER 100%"
280 IF R=100 THEN 90
290 GOTO 210
300 IF M-.01<M1 THEN 320
310 GOTO 250
320 R=INT(R*100+.5)/100
330 PRINT "INTEREST RATE IS";R;"%"
340 PRINT
350 INPUT "WANT TO DO ANOTHER (Y/N)?  "; A$
360 A$=A$(1,1)
370 IF A$="Y" THEN 90
380 IF A$<>"N" THEN 350
390 END
```

Program listing.

Add Handshaking to Apple's High Speed Interface

This modification to the Apple Serial Card allows it to handshake properly with other devices such as printers. The mod is unique because it does not require a software patch.

Jeffrey G. Mazur
8041 Sadring Ave.
Canoga Park, CA 91304

By now, most people who own an Apple High Speed Serial Interface are aware of its lack of handshaking capabilities. Several modifications have appeared (one directly from Apple), but these involve using the data input line as a CTS (clear to send) or DTR (data terminal ready) input.

A major problem—aside from giving up duplex operation—with this approach is that it requires a software patch. This means that every time the serial card is to be used, a patch program must be loaded from cassette or disk. Also, the patch restricts the card to one particular slot. And if that isn't enough, try loading the patch from Apple Pascal!

These problems defeat the purpose of having "intelligent interfaces" with complete software in PROM. Fortunately, there is a simple modification that can be added to perform the handshaking without any software patches.

The schematic for the modification is shown in Fig. 1. Note that no direct connection to the serial card is required; thus, the entire modification could be done on a prototyping board and plugged into any slot on the Apple II. This would avoid warranty problems incurred with making changes to the serial card itself.

However, the low parts count (two, with TTL handshaking) of this circuit makes it feasible to mount everything and make all connections directly on the serial card. This is the preferred method, since it doesn't take up another slot and saves the price of the protoboard.

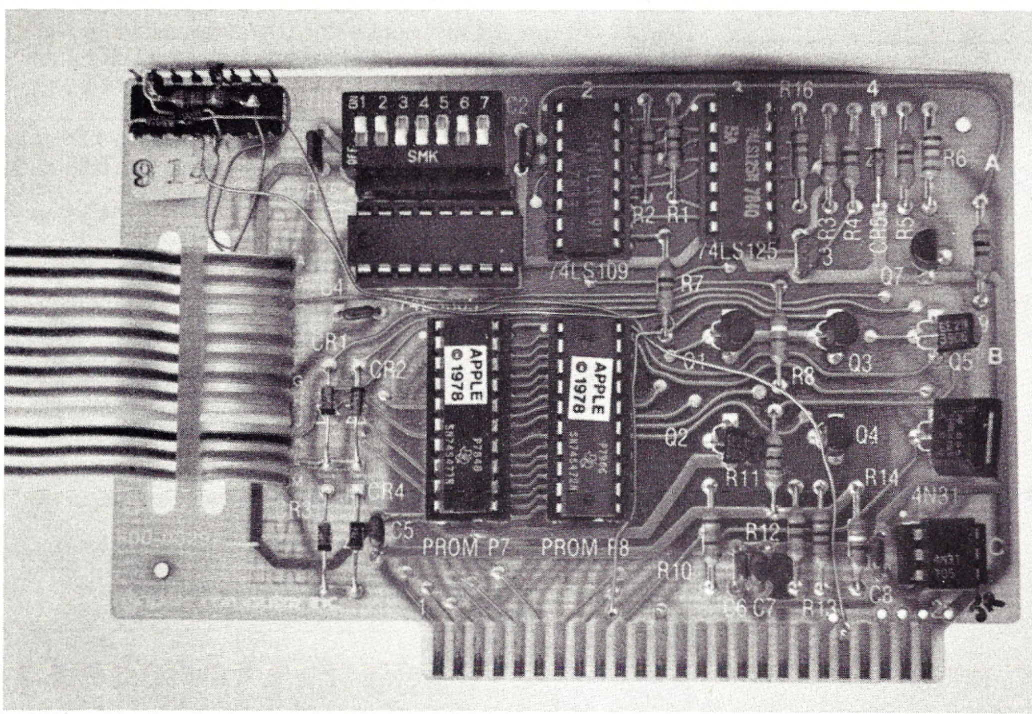


Photo 1. Serial Card with modification on component side. Note all parts mount directly on IC.

Theory

This simple handshaking circuit makes use of the RDY (ready) signal on the 6502 and

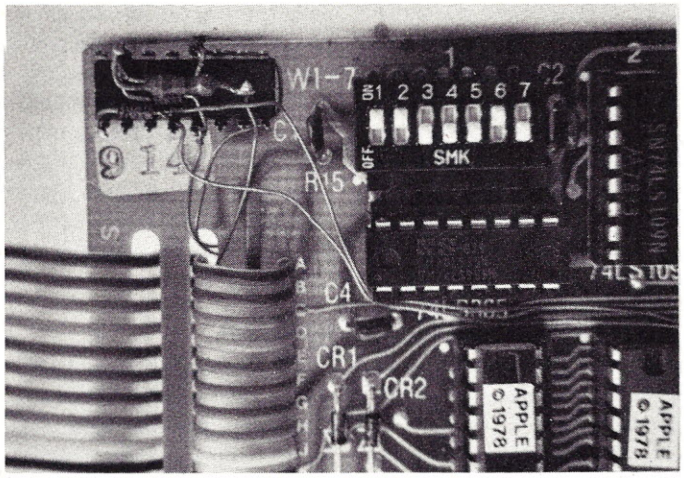


Photo 2. Close-up of added parts.

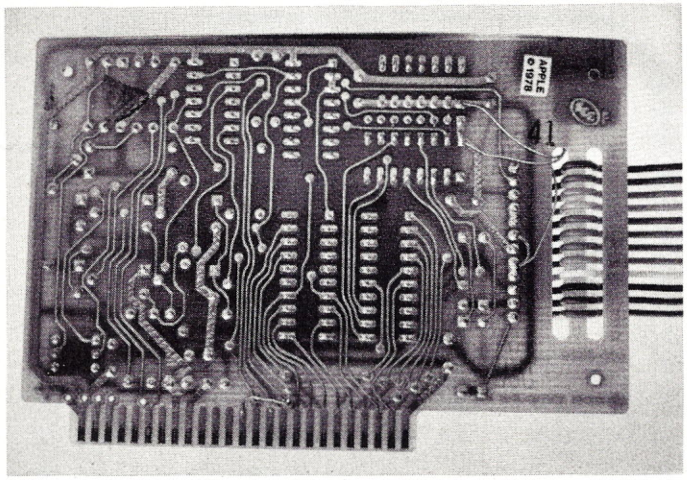


Photo 3. Back side of boards showing connections to ground, + 5 V, DB-25 and pin 38 of edge connector (see Fig. 2).

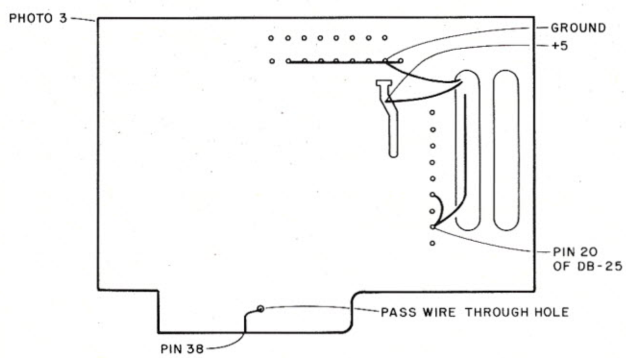


Fig. 2. Points in Photo 3.

Apple bus. This signal is normally high (via a pull-up resistor); when pulled low it causes the 6502 to enter a wait state. The computer will do nothing until the RDY line goes high again.

This is exactly what we want the computer to do when a printer or other serial device signals (via a low CTS or DTR) that it is busy or cannot accept any more data into its buffer. If the 6502 did not restrict when the RDY line could change state this modification would consist simply of one diode! However, the specifications on the 6502 state that the RDY line should change from high to low or vice versa only during the Ø1 clock time (if it changes during Ø2 the CPU will hang up).

This is one of my pet peeves about the 6502: why didn't they include a latch on the chip to ensure this? Since they didn't, we must use one section of a 74LS75 quad data latch to restrict the RDY line to change only during Ø1. D1 is used to couple only the low state of the latch to the RDY line on the Apple bus. This is necessary to allow other peripheral boards to also use the RDY line (i.e., pull it low) without conflicting with the high state of IC1. Finally, R1, D2 and D3 are used (if necessary) to convert RS-232 levels to TTL levels.

Construction

If you want to use a separate protoboard, simply mount the parts and connect the CTS or DTR line from the printer to pin 2 of IC1 or R1. Mounting the modification on the serial board can be done in one of two ways.

A socket for IC1 can be added by drilling six holes in the upper left-hand corner of the board. All unused pins of the socket should be cut off. Then wire on the back side of the circuit board using small gauge (e.g., #30 wire-wrap) wire.

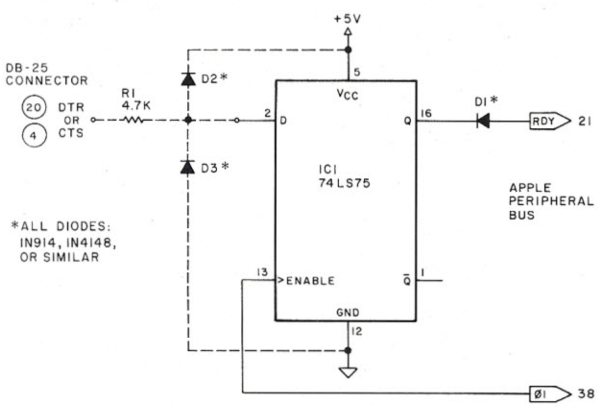


Fig. 1. Simple handshaking circuit for the Apple Serial Card. Marked components are used if handshaking signal uses RS-232 levels. If your device uses an active high busy signal, move D1 to IC1, pin 1.

The other method involves mounting the parts on the component side. This method, which requires no drilling, is outlined below.

Refer to Photo 1 for the location of IC1 and carefully glue the chip upside down on the PC board. Make sure that the glue has dried securely before continuing with the wiring. Now carefully connect IC1 as neatly as possible to avoid shorts or bad connections.

When attaching wires to the IC pins, strip back only a small amount of insulation and then tin the pin and the wire with solder. Holding the wire vertically, lightly touch the iron to the IC pin and then quickly join the wire and remove the iron. Use similar care when attaching the other parts to IC1. Note that the ground connection can be made at the common trace of the DIP switches. The fat trace just below it is +5 V. If you have any doubts about your soldering capabilities, you can use insulating spaghetti or shrink tubing to further protect against shorts.

When wiring to the connector fingers, use the smallest amount of solder possible and make the connection at the top of the finger. Avoid getting any solder on the lower two-thirds of the finger (the part that goes into the computer).

Finally, to make connection to the DB-25 cable, determine which pin carries your printer's busy signal. If it uses a CTS signal, it will probably be on pin 4. Therefore, attach the wire from IC1, pin 2 (TTL), or R1 (RS-232) to hole D (yellow wire of the ribbon cable). If your printer uses a DTR signal (pin 20), connect to hole H (gray wire). Now check all your wiring and—for a professional job—use tiny drops of silicone sealer to secure the wires to the circuit board.

Testing and Operation

When you are sure that everything is correct, plug the serial

board back into the computer. Connect your printer and go through the normal procedure to check it out. If the computer hangs up when you turn it on, check the polarity of the busy signal from the printer. Most devices use an active low signal (i.e., line is normally high; a low level signifies that the device is not ready to receive more data).

If your printer works in reverse, move D1 from IC1, pin 16, to pin 1. This picks up the inverted output from the latch.

If you have been running the printer at a slow baud rate to avoid overflow problems, check it out first at this slow rate. If it operates normally, change both the serial card and printer baud rates to the maximum, keeping them both the same. Now watch the printer zip away.

If your printer has an input buffer, the Apple will now send data to the buffer at a high baud rate (usually filling the buffer faster than the printhead can empty it). If the buffer becomes full, it will pull the RDY line low, halting the computer while the printer catches up. At this point the computer will sync up, with the printer sending one character for each character the printhead prints (but not the same one). After the computer stops sending, the printer will continue to operate until it empties the buffer.

Conclusion

Since most of the newer, low-cost printers have input buffers, being able to handshake correctly is an important feature. Although the Apple Serial Card lacks this capability as supplied, there are simple ways to add this feature.

The modification I have presented here seems to be the best approach, since it does not involve any software patches. I have used the serial card in this manner for many months (even with Pascal) without any problems. With hard copy as relatively slow as it is, it's nice to know that the printer is working as fast as it can. ■

Blazimir P. Mise
Bulevar Oktob, Revolucije 82
Yu-11040 Belgrade, Yugoslavia

Predict Variable Trends

Interpolation by microcomputer is a practical alternative.

There are many times when we have to predict the value of an unknown function F(X) or a variable such as voltage, current, capacitance, impedance, etc., for an arbitrary value of an independent variable X, based on a group of known interpolation points $(X_i, F(X_i))$, which are the result of a finite number of tests or measurements at those points.

Hence, it is possible to monitor, for instance, temperature, pressure or radiation trends as a function of time and to predict their value using some kind of "predicting" algorithm. For example, this method might have been very useful at Three Mile Island on March 28, 1979!

Lagrange's Interpolation Formula enables approximate evaluation of any function, at an arbitrary point, based on a known group of interpolation points. Lagrange's formula allows us to predict the value of any physical, chemical or electrical parameter (which could be monitored by means of some kind of transducer, converted by an A/D converter, and then processed by a microcomputer using our "predicting" algorithm), and perhaps prevent system failures before they can occur.

The program listing was written originally for an SWTP 6800 microcomputer with 16K of memory (9K occupied by BASIC Interpreter), which utilizes Lagrange's Interpolation Formula to predict the value of any input variable X. In this example, port 1 is assigned to terminal, while port 7 is assigned to printer.

The sample run was performed with five interpolation data points; abscissae were angles in degrees (0, 30, 45, 60, 90°). According to well-known values of sine function for those angles (0, 0.5, 0.707, 0.866, 1, respectively), the computer predicted almost the exact value of the variable (sine function in our case) for an arbitrary data point X (here, it is angle of 20 degrees). Note that the error rate was less than 10E(−5) or 0.001 percent.

User-defined and decoded control characters at lines 1 and

```
1 PRINT CHR$(16);CHR$(22)
5 DIM X(20);Y(20)
7 REM * CREATED BY BLAZIMIR P. MISE, EE *
8 REM * @ SWTP 6800 COMPUTING SYSTEM *
10 INPUT " NUMBER OF DATA POINTS ",N
12 PRINT
15 FOR I=1 TO N
20 PRINT " * DATA POINT # ";I
28 INPUT "  ABSCISSAE X(I)",Z : X(I)=Z
30 INPUT "  DATA VALUE    ",Z1 : Y(I)=Z1
32 PRINT
35 NEXT I
40 INPUT "  DESIRED DATA POINT X",X
45  S=0
50 FOR I=1 TO N
55  P=1
60 FOR J=1 TO N
65 IF J=1 THEN 100
70  A=X-X(J)
75  B=X(I)-X(J)
80  P=P*A/B
100 NEXT J
105  S=S+Y(I)*P
110 NEXT I
112 PRINT
113 PRINT CHR$(16);CHR$(22)
114 PORT= 7
115 PRINT "   LAGRANGE'S INTERPOLATION FORMULAE"
116 PRINT "   FOR N RANDOM DATA POINTS"
117 PRINT "               VERSION 1.0"
118 PRINT
120 PRINT " DATA POINT NO.  ABSCISAE  DATA VALUE"
122 PRINT
125 FOR I=1 TO N
127 PRINT TAB(5);I;TAB(20);X(I);TAB(30);Y(I)
130 NEXT I
132 PRINT
133 PRINT " DESIRED DATA POINT X= ";X
134 PRINT
135 PRINT " POLYNOMIAL'S VALUE"
137 PRINT " P(X)= ";S
140 PRINT
142 PRINT " CREATED BY BLAZIMIR P. MISE, EE"
145 PRINT " @ SWTP 6800 COMPUTING SYSTEM"
147 PORT= 1
150 END
```

Program listing.

113 with the basic CHR$ function make efficient terminal commands to cursor home (ASCII CNTRL-P, DLE, 16) and erase screen (ASCII CNTRL-V, SYN, 22).■

```
LAGRANGE'S INTERPOLATION FORMULAE
 FOR N RANDOM DATA POINTS
             VERSION 1.0

DATA POINT NO.  ABSCISAE  DATA VALUE

   1               0          0
   2               30         0.5
   3               45         0.707
   4               60         0.866
   5               90         1

DESIRED DATA POINT X= 20

POLYNOMIAL'S VALUE
P(X)= 0.341999988

CREATED BY BLAZIMIR P. MISE, EE
@ SWTP 6800 COMPUTER SYSTEM

COMPARE RESULT WITH EXACT VALUE OF
SINE FUNCTION :



0.341996


ERROR :
3.88E-06
```

Sample run.

Use Your Exidy As a Smart Terminal

"Talk" to maxi-computers with the Exidy Sorcerer.

Ernest E. Bergmann
Physics, Bldg. #16
Lehigh University
Bethlehem, PA 18015

The Exidy Sorcerer, which costs about the same as a dumb computer terminal, can be used as a 300 baud dial-up terminal to a large computer (such as the DEC-20 at our college campus). I have developed software that successfully uses the Sorcerer's on-board RS-232-compatible serial port. Not only can I talk to the large computer with my "terminal," but I can transfer larger data files (20K ASCII files, such as the assembly listing) between my computer and "big brother."

With the DEC-20's help, I can now print, edit, maintain and exchange large files with other computers for the cost of a modem. I wrote the manuscript for this article with the help of a large computer that has well-maintained software (editors and formatters) and hardware (disk storage and printers).

My hardware consists of the Exidy Sorcerer, two cassette recorders, one modified TV monitor, a modem and a telephone. The bare Sorcerer has quite a few facilities on-board, including the necessary RS-232 hardware to connect to the modem. However, it took me a while to figure out how to use the serial interface because of limited documentation and what appears to be a sofware bug in the monitor.

The Exidy's UART function can be selected in software to communicate either with the cassette recorders or through the RS-232 interface. The choice is controlled by bit 7, the most significant bit of port FE (hex). For operation of the RS-232 interface this bit must always be set (on). Whenever this bit is reset (off), the cassette interface has the UART's attention and, unfortunately for this application, the RS-232 output then has a space signal (the opposite signal, a mark, must be used instead between transmitted characters).

This same output port FE is also used to control the baud rate with bit 6 (on for 1200 baud and off for 300 baud). The motors for cassette #1 and #2 are controlled by bits 4 and 5; the remaining output bits of port FE, 0 through 3, are used to scan the keyboard. The monitor's keyboard scan routine, which normally sees heavy use, needs to constantly change these four, lowest-order bits of port FE. Inadvertently, this keyboard routine is always resetting bits 5 through 7, rendering the RS-232 unusable.

To correct this software bug in the monitor ROM, I hit upon the following solution: copy the keyboard scan routine in its entirety into RAM but modify the few places where the offending bits are produced. In this way, I wrote the subroutine KEYBD2 (more suited to my application), which appears in the program listing, lines 40 through 179. Line 46 differs from the original keyboard scan routine, which had "LD A,1"; I changed line 64 in a similar way.

There is a second keyboard routine in the monitor, QCKCHK, which quickly checks to see if a control-C, Escape or RUN/STOP is pressed. It, too, must use the output port FE, but it can leave the most significant bits in the state specified by bytes located at IY+45H and IY+3DH in memory. Thus, QCKCHK does not need to be replaced; changing these two bytes is simpler, see lines 225 and 226.

Full-Duplex Operation

Full-duplex means that our keyboard's output is sent to the modem and, simultaneously, the modem's received signal must be displayed on the TV monitor. This simultaneous I/O means we must be always listening to both the modem and the keyboard and talking to both the CRT and the modem. This may seem easy to do with an interrupt-driven computer, but the Sorcerer's on-board hardware does not seem designed that way.

We may envision the problem as two separate processes:

1. If a key is pressed, send its value out the RS-232.
2. If a character is received from the modem, place it on the screen.

Of the two, the second process is faster and is the basis of the routine INPIFR, "input if ready." This subroutine (starting at line 180) checks the UART status, and if a character has been received, it transfers control to SHOW (line 186), which puts the character on the screen. Placing INPIFR at the end of KEYBD2 and CALL INPIFR elsewhere in the program appears to catch all of the input character stream without interfering with process 1 which can now be thought of as:

1. When a key is pressed, send its value out the RS-232.

The top level of my program is TERMNL (lines 223 to 232); it is the embodiment of the modified process 1. It calls KEYBD2 until a character is typed. SEROUT (the RS-232 output driver) is called when a character is eventually typed. If line 230 and all lines beyond 232 were removed, then you would have a complete, full-duplex, dumb terminal.

Adding SMARTS

I still needed to develop more versatile software that would take advantage of the Sorcerer's programmable intelligence. I have implemented four principal objectives:

1. To send a large ASCII file from the Sorcerer's memory to the computing center.
2. To receive a large ASCII file from the computing center and place it in my RAM.
3. To view the contents of RAM.
4. To exit to the monitor to do local chores such as load from or save to cassette.

I figured that I would never need to send graphic characters (here the Exidy sets the most significant bit to one) to the computer center; in fact, the central computer ignores the parity bit. Therefore, I could filter out such characters (using lines 229 and 230) before sending them to SEROUT.

I arbitrarily chose the graphic-shift-period combination, 0EFH, as a special-graphics character to signify that I desired one of the four special functions. Any other graphic character is ignored (lines 234 and 235). When this special character is typed, a prompt, <<< appears on the screen, indicating that it is ready for one of the four types of requests.

Rather than writing my own routines to fill a line buffer with the request and then routines to interpret it, I was able to make use of routines already present in the monitor's ROM. The monitor's input routine, MINPUT, normally makes use of the keyboard scan routine that contains the software bug we need to avoid, so I patched the input with KEYBD2 at memory locations IY + 41H and IY + 42H (lines 236 to 239).

Starting with line 244, I used MINPUT, set HL to the beginning of the line buffer and called FTOKEN, which will return with the Z-flag set if the line is empty; otherwise, register A will contain the first non-blank character in the line, which should be S, R, V or X for the four possible commands. If none of these four characters is found, the program will display questions marks and ignore the request (QQ, starting at line 257).

Sending Files

Before I tell you about the software for sending files, I will describe its use. Suppose I need to send several kilobytes of text that is sitting in the Sorcerer's RAM starting at location 1000H. Assuming that the central computer is ready to receive it, I type the special-graphics character (which produces the prompt) and then S, <space>, 1000 and <CR>.

The "terminal" will start sending the contents of RAM until either a key that QCKCHK can detect is pressed or the end-of-file is reached. When the "terminal" stops sending, it will display the address in memory of the byte beyond the last one sent. (So I can easily resume sending where I left off or can estimate the length of the file just sent.) The end-of-file character that I use in RAM, MEOF, is defined in line 34.

Writing a suitable routine is challenging because of the inability of the large computer to keep up with a long, continuous, input stream. Undoubtedly, each computer system is likely to react differently.

The TOPS-20 operating system of the DEC-20 computer reacts in the following way: initially it accepts input, gradually falling behind in the echoed signal (sending back what it receives so that it is visible on my TV monitor). After it has fallen behind about three lines (100 to 200 characters), it sends out a control-S, which is supposed to mean that I should pause in my transmission while the DEC-20 catches up by processing its input buffer. A little later, it sends a control-Q to indicate that it is OK for my com-

Program listing.

```
ADDR   OBJECT    ST

                 0001
                 0002 ;
                 0003 ;FULL DUPLEX ROUTINES FOR THE EXIDY
                 0004 ;***by E.E.Bersmann, Jan, 1980*********
>E1A2            0005 GETIY    EQU     0E1A2H  ;MONITOR ADDRESS OF
                 0006                          ;ROUTINE THAT PLACES
                 0007                          ;INITIAL STACK TOP
                 0008                          ;LOCATION IN IY
>E00C            0009 SEND     EQU     0E00CH  ;MONITOR ROUTINE THAT
                 0010                          ;SENDS TO CURRENT OUT-
                 0011                          ;PUT DEVICE
>006C            0012 LSTKEY   EQU     6CH     ;LAST KEY
                 0013 ;START OF TABLE LOCATIONS FOR...
>EC1E            0014 INSTBL   EQU     0EC1EH  ;INSTRUCTIONS
>EC6E            0015 GRATBL   EQU     0EC6EH  ;GRAPHICS
>ECBE            0016 CONTBL   EQU     0ECBEH  ;CONTROL
>ED0E            0017 SHITBL   EQU     0ED0EH  ;SHIFT
>ED5E            0018 SLOTBL   EQU     0ED5EH  ;SHIFT-LOCK
>EDAE            0019 UNSTBL   EQU     0EDAEH  ;UNSHIFT
>000D            0020 CR       EQU     0DH
>000A            0021 LF       EQU     0AH
                 0022 ;*************************************
                 0023 ;ADDITIONAL EQUS FOR SMARTS:
>00FF            0024 VDLY     EQU     0FFH    ;VIEW DISP RATE
>E015            0025 QCKCHK   EQU     0E015H
>E205            0026 CRLF     EQU     0E205H
>E13A            0027 MINPUT   EQU     0E13AH
>E1BA            0028 SNDLIN   EQU     0E1BAH
>E225            0029 FTOKEN   EQU     0E225H
>E1E8            0030 MUNCNV   EQU     0E1E8H
>E22F            0031 GTOKN2   EQU     0E22FH
>E23D            0032 MCONVA   EQU     0E23DH
>FF01            0033 OUTDLY   EQU     0FF01H  ;DELAY FOR BSOUT
>0000            0034 MEOF     EQU     00      ;EOF IN RAM
>001A            0035 CCEOF    EQU     01AH    ;EOF USED BY
                 0036                          ;COMP CENTER
                 0037 ;************************************
                 0038          GLOBAL  DUPLEX
'0000  C35601'   0039 DUPLEX   JP      TERMNL
'0003  FDE5      0040 KEYBD2   PUSH    IY
'0005  CDA2E1    0041          CALL    GETIY
'0008  C5        0042          PUSH    BC
'0009  D5        0043          PUSH    DE
'000A  E5        0044          PUSH    HL
'000B  DDE5      0045          PUSH    IX
'000D  3E81      0046          LD      A,81H
'000F  D3FE      0047          OUT     (0FEH),A
'0011  DBFE      0048          IN      A,(0FEH)
'0013  CB4F      0049          BIT     1,A
'0015  200E      0050          JR      NZ,NORPT-$
'0017  018813    0051          LD      BC,5000
'001A  0B        0052 REPET    DEC     BC
'001B  78        0053          LD      A,B
'001C  B1        0054          OR      C
'001D  20FB      0055          JR      NZ,REPET-$
'001F  FD7E6C    0056          LD      A,(IY+6CH)
'0022  C3F700'   0057          JP      FINEND
'0025  AF        0058 NORPT    XOR     A
'0026  0EFE      0059          LD      C,0FEH
'0028  5F        0060          LD      E,A
'0029  CBFB      0061          SET     7,E
'002B  1600      0062 MLOOP    LD      D,0
'002D  42        0063          LD      B,D
'002E  2680      0064          LD      H,80H
'0030  DD211EEC  0065          LD      IX,INSTBL
'0034  ED61      0066 SLOOP    OUT     (C),H
'0036  2E01      0067          LD      L,1
'0038  ED78      0068 BLOOP    IN      A,(C)
'003A  A5        0069          AND     L
'003B  C2C900'   0070          JP      NZ,ABIT1
'003E  E5        0071          PUSH    HL
'003F  210001    0072          LD      HL,100H
'0042  2D        0073 DEBOUN   DEC     L
'0043  20FD      0074          JR      NZ,DEBOUN-$
'0045  25        0075          DEC     H
'0046  20FA      0076          JR      NZ,DEBOUN-$
'0048  E1        0077          POP     HL
'0049  ED78      0078          IN      A,(C)
'004B  A5        0079          AND     L
'004C  C2C900'   0080          JP      NZ,ABIT1
'004F  DD5600    0081          LD      D,(IX+0)
'0052  CB7A      0082          BIT     7,D
'0054  281A      0083          JR      Z,CODED-$
'0056  CB62      0084          BIT     4,D
'0058  2802      0085          JR      Z,NONGRA-$
'005A  CBF0      0086          SET     6,B
'005C  CB5A      0087 NONGRA   BIT     3,D
'005E  2802      0088          JR      Z,NONCON-$
'0060  CBE8      0089          SET     5,B
'0062  CB52      0090 NONCON   BIT     2,D
'0064  2802      0091          JR      Z,NONSHI-$
'0066  CBE0      0092          SET     4,B
'0068  CB4A      0093 NONSHI   BIT     1,D
'006A  285D      0094          JR      Z,ABIT1-$
'006C  CBD8      0095          SET     3,B
'006E  1859      0096          JR      ABIT1-$
'0070  E5        0097 CODED    PUSH    HL
'0071  D5        0098          PUSH    DE
'0072  DDE5      0099          PUSH    IX
'0074  E1        0100          POP     HL
'0075  111EEC    0101          LD      DE,INSTBL
'0078  B7        0102          OR      A
'0079  ED52      0103          SBC     HL,DE
                 0104 ;DECIDE WHICH TABLE TO USE
'007B  D1        0105          POP     DE
'007C  CB70      0106          BIT     6,B
'007E  2815      0107          JR      Z,NOGRAP-$
'0080  CB72      0108          BIT     6,D
'0082  2811      0109          JR      Z,NOGRAP-$
'0084  D5        0110          PUSH    DE
'0085  116EEC    0111          LD      DE,GRATBL
'0088  19        0112          ADD     HL,DE
'0089  7E        0113          LD      A,(HL)
```

puter to send more.

When I wrote a routine that used the control-S and control-Q characters to permit the DEC-20 to catch up, the latter would occasionally produce a <bell> (an ASCII 7), and subsequent investigation showed that characters had been lost. There is a bug in the DEC-20 software that causes it to send the control-S too late!

My current approach has proven much more successful and should work for almost any computer installation. I never let the echo fall more than one line behind. During block transmissions, I use BSOUT (lines 337 to 363) instead of SEROUT (lines 191 to 201). BSOUT acts just like SEROUT except that when it must transmit a <CR>, it does not return to the calling program (lines 349 to 351) but proceeds to LWAIT, which waits for a <LF> before allowing a return to the program that called BSOUT.

If LWAIT merely did nothing until a <LF> came echoing back, then we would miss the display of the last few characters in each line. So LWAIT takes every character it receives and forwards it to FSEND, which behaves much like the send in the monitor's ROM except that it filters out control characters for special attention. The <CR> and <LF> cause no problems for SEND; they are not filtered out.

Thus far I have received <null>s (usually at the beginning of a new line) and <bell>s. I have written FSEND to ignore the former and display "<BELL>" for the latter (using lines 384 through 396). Since I need to be aware of any other possible control codes, FSEND will type out the hexadecimal value of any such control character and follow it with four equal signs for an eye-catching effect (lines 376 through 383).

If the block transmission is terminated because QCKCHK has detected a keypress, the next RAM address that was to be used will be displayed. The rest of the line on the TV monitor will be filled with >s so that I can quickly locate this information. (These tasks are done by ADONE, starting at line 266.)

Otherwise, the block transmission will terminate when the MEOF character is found in RAM. Control is transferred to OUTEND (line 289), which sends the character CCEOF (defined at line 35), which is a control-Z for the DEC-20 system. This character signifies to the large computer that the file is to be closed. After the CCEOF has been output, control goes to ADONE.

When I started writing the block output routine BOUT, I did not realize how much effort it would require. Nevertheless, I am pleased with the result; now I can burden the large computer with large files, and have confidence that they will be received intact.

Receiving Files

To use this feature to load a file from the large computer to my RAM starting at location 1800H, for example, I would type a command to the large computer that would cause it to type the file without pause on my "terminal," but I would leave out the <CR>. That completes the command. Most computer systems will not act upon a command until they receive this <CR>. I then type the special-graphics character, an R, a space and the address 1800. The screen will display
@TYPE LIST7<<<R 1800
The first part of the line depends upon the commands that the large computer recognizes and the name of the file you want.

If everything appears in order, I then type a <CR>. My program then figures out what I want and sends a <CR> to the computing center. The DEC-20 then proceeds to type LIST7 (or whatever file I chose), which then appears on my display as well as being placed in my RAM.

When typing is complete or I do not want any more, I hit the key RUN/STOP. My program will then place a <CR>, a <LF> and the MEOF into RAM to complete what it has recorded. Control then passes to ADONE, which will show the address in memory of the next available byte.

Writing the block input routine, BINP (lines 293 through 321), was straightforward, but it was necessary to filter the incoming stream of characters to eliminate the <null>s (which happen to be my MEOF) before they could enter RAM. Otherwise, my files in RAM got all chopped up!

Viewing Files

My program permits examination of ASCII files in RAM without interaction with the computer center. After you type the special-graphics character, V, space, a hexadecimal address and a <CR>, the ASCII file will be displayed on the monitor at a rate of about 100 characters per second. As with sending files, the display will terminate with MEOF found in RAM or upon QCKCHK's seeing a keypress. Upon termination, a hexadecimal address will be displayed.

I had no problems writing VIEW (lines 202 through 221). I must emphasize that the display rate is determined by VDLY (defined in line 24 and used in line 209).

Implementation

With the aid of the accompanying program listing, you should be able to place the program in your Sorcerer even without the aid of an assembler. I wrote the program with this aim in mind by providing for the selection of various options without the need for reassembly.

This is a relocatable assembly listing. To relocate the program, just add the address of another location to all addresses in the listing that have a single quote sign (or apostrophe) next to them. With this in mind, it is a simple matter to use the (EN)TER command of the Sorcerer's monitor to put the program in service.

First, you should try to set the "dumb" functions to operate properly. SEROUT is currently written to generate even parity

```
'008A  CBFF            SET   7,A
'008C  D1              POP   DE
'008D  CB60            BIT   4,B
'008F  2826            JR    Z,FINOP-$
'0091  CBF7            SET   6,A
'0093  1822            JR    FINOP-$
'0095  D5       NOGRAP PUSH  DE
'0096  CB68            BIT   5,B
'0098  2805            JR    Z,SKIP1-$
'009A  11BEEC          LD    DE,CONTBL
'009D  1815            JR    SKIP4-$
'009F  CB60     SKIP1  BIT   4,B
'00A1  2805            JR    Z,SKIP2-$
'00A3  110EED          LD    DE,SHITBL
'00A6  180C            JR    SKIP4-$
'00A8  CB58     SKIP2  BIT   3,B
'00AA  2805            JR    Z,SKIP3-$
'00AC  115EED          LD    DE,SLOTBL
'00AF  1803            JR    SKIP4-$
'00B1  11AEED   SKIP3  LD    DE,UNSTBL
'00B4  19       SKIP4  ADD   HL,DE
'00B5  D1              POP   DE
'00B6  7E              LD    A,(HL)
'00B7  CBE3     FINOP  SET   4,E
'00B9  CBDB            SET   3,E
'00BB  CBD3            SET   2,E
'00BD  CBBB            RES   7,E
'00BF  E1              POP   HL
'00C0  F5              PUSH  AF
'00C1  ED78     WAITK  IN    A,(C)
'00C3  A5              AND   L
'00C4  28FB            JR    Z,WAITK-$
'00C6  F1              POP   AF
'00C7  180B            JR    BITEND-$
'00C9  CB05     ABIT1  RLC   L
'00CB  3E20            LD    A,20H
'00CD  BD              CP    L
'00CE  2002            JR    NZ,SKIP5-$
'00D0  CBD3            SET   2,E
'00D2  DD23     SKIP5  INC   IX
'00D4  CB53     BITEND BIT   2,E
'00D6  CA3800'         JP    Z,BLOOP
'00D9  CB93            RES   2,E
'00DB  CB5B            BIT   3,E
'00DD  2007            JR    NZ,SECEND-$
'00DF  24              INC   H
'00E0  3E90            LD    A,90H
'00E2  BC              CP    H
'00E3  C23400'         JP    NZ,SLOOP
'00E6  CB9B     SECEND RES   3,E
'00E8  37              SCF
'00E9  CB7B            BIT   7,E
'00EB  2803            JR    Z,SKIP6-$
'00ED  AF              XOR   A
'00EE  CBE3            SET   4,E
'00F0  CB63     SKIP6  BIT   4,E
'00F2  2003            JR    NZ,FINEND-$
'00F4  C32B00'         JP    MLOOP
'00F7  DDE1     FINEND POP   IX
'00F9  E1              POP   HL
'00FA  D1              POP   DE
'00FB  C1              POP   BC
'00FC  B7              OR    A
'00FD  2803            JR    Z,KEYRET-$
'00FF  FD776C          LD    (IY+LSTKEY),A
'0102  FDE1     KEYRET POP   IY
'0104  F5       INPIFR PUSH  AF
'0105  DBFD            IN    A,(OFDH)
'0107  CB4F            BIT   1,A
'0109  2002            JR    NZ,SHOW-$
'010B  F1              POP   AF
'010C  C9              RET
'010D  DBFC     SHOW   IN    A,(OFCH)
'010F  CBBF            RES   7,A
'0111  CD0CE0          CALL  SEND
'0114  F1              POP   AF
'0115  C9              RET
'0116  F5       SEROUT PUSH  AF
'0117  DBFD     WBE    IN    A,(OFDH)
'0119  CB47            BIT   0,A
'011B  CD0401'         CALL  INPIFR
'011E  28F7            JR    Z,WBE-$
'0120  F1              POP   AF
'0121  A7              AND   A
'0122  EA2701'         JP    PE,OUTOK
'0125  CBFF            SET   7,A
'0127  D3FC     OUTOK  OUT   (OFCH),A
'0129  C9              RET
'012A  CDC601'  VIEW   CALL  GETADR
'012D  2879            JR    Z,QQ-$
'012F  CD15E0   BVWLP  CALL  QCKCHK
'0132  C2BE01'         JP    NZ,ADONE
'0135  1A              LD    A,(DE)
'0136  13              INC   DE
'0137  CBBF            RES   7,A
'0139  06FF            LD    B,VDLY
'013B  CD0401'  VDLYLP CALL  INPIFR
'013E  10FB            DJNZ  VDLYLP-$
'0140  FE00            CP    MEOF
'0142  287A            JR    Z,ADONE-$
'0144  FE0D            CP    CR
'0146  2809            JR    Z,VCRLF-$
'0148  FE0A            CP    LF
'014A  28E3            JR    Z,BVWLP-$
'014C  CD0CE0          CALL  SEND
'014F  18DE            JR    BVWLP-$
'0151  CD05E2   VCRLF  CALL  CRLF
'0154  18D9            JR    BVWLP-$
                ;**********MAIN ENTRY********
'0156  CDA2E1   TERMNL CALL  GETIY
'0159  3E80            LD    A,80H
'015B  FD7745           LD    (IY+45H),A
'015E  FD773D           LD    (IY+3DH),A
'0161  CD0300'         CALL  KEYBD2
'0164  28F0            JR    Z,TERMNL-$
'0166  CB7F            BIT   7,A
'0168  2005            JR    NZ,SMARTS-$
'016A  CD1601'         CALL  SEROUT
'016D  18E7            JR    TERMNL-$
'016F  CDA2E1   SMARTS CALL  GETIY
```


✔ *Reader Service index—page 241*

```
'0172  FEEF      0234          CP    0EFH      ;GS'.'
'0174  20E0      0235          JR    NZ,TERMNL-$
'0176  210300'   0236          LD    HL,KEYBD2        ;PATCH
'0179  FD7541    0237          LD    (IY+41H),L       ;MONOTOR
'017C  FD7442    0238          LD    (IY+42H),H       ;INPUT
'017F  FD364300  0239          LD    (IY+43H),0       ;NO BATCH
'0183  3E3C      0240          LD    A,'<'            ;PROMPT
'0185  0603      0241          LD    B,3              ;WITH 3
'0187  CD0CE0    0242 PLP      CALL  SEND             ;'<'
'018A  10FB      0243          DJNZ  PLP-$
'018C  CD3AE1    0244          CALL  MINPUT
'018F  FDE5      0245          PUSH  IY
'0191  E1        0246          POP   HL
'0192  CD25E2    0247          CALL  FTOKEN
'0195  2819      0248          JR    Z,DONE-$;PREMATURE<CR>
'0197  FE53      0249          CP    'S'
'0199  2832      0250          JR    Z,BOUT-$
'019B  FE52      0251          CP    'R'
'019D  2859      0252          JR    Z,BINP-$
'019F  FE56      0253          CP    'V'
'01A1  2887      0254          JR    Z,VIEW-$
'01A3  FE58      0255          CP    'X'
'01A5  CA00E0    0256          JP    Z,0E000H;COLD START
'01A8  3E3F      0257 QQ       LD    A,'?'
'01AA  CD0CE0    0258          CALL  SEND
'01AD  CD0CE0    0259          CALL  SEND
'01B0  3E3E      0260 DONE     LD    A,'>'
'01B2  0640      0261          LD    B,64
'01B4  CD0CE0    0262 DONELP   CALL  SEND
'01B7  10FB      0263          DJNZ  DONELP-$
'01B9  CD05E2    0264          CALL  CRLF
'01BC  1898      0265          JR    TERMNL-$
'01BE  CD05E2    0266 ADONE    CALL  CRLF
'01C1  CDE8E1    0267          CALL  MUNCNV
'01C4  18EA      0268          JR    DONE-$
'01C6  CD2FE2    0269 GETADR   CALL  GTOKN2
'01C9  C8        0270          RET   Z
'01CA  C33DE2    0271          JP    MCONVA
'01CD  CDC601'   0272 BOUT     CALL  GETADR
'01D0  28D6      0273          JR    Z,QQ-$
'01D2  CD15E0    0274 BOUTLP   CALL  QCKCHK
'01D5  20E7      0275          JR    NZ,ADONE-$
'01D7  1A        0276          LD    A,(DE)
'01D8  13        0277          INC   DE
'01D9  0101FF    0278          LD    BC,OUTDLY
'01DC  CD2F02'   0279 SDLYLP   CALL  BINIFR
'01DF  10FB      0280          DJNZ  SDLYLP-$
'01E1  0D        0281          DEC   C
'01E2  20F8      0282          JR    NZ,SDLYLP-$
'01E4  FE0A      0283          CP    LF
'01E6  28EA      0284 LFDFT    JR    Z,BOUTLP-$;SKIP
'01E8  FE00      0285          CP    MEOF
'01EA  2805      0286          JR    Z,OUTEND-$
'01EC  CD4102'   0287          CALL  BSOUT
'01EF  18E1      0288          JR    BOUTLP-$
'01F1  3E1A      0289 OUTEND   LD    A,CCEOF
'01F3  CD4102'   0290          CALL  BSOUT
'01F6  18C6      0291          JR    ADONE-$
                 0292 ;
'01F8  CDC601'   0293 BINP     CALL  GETADR
'01FB  28AB      0294          JR    Z,QQ-$
'01FD  3E0D      0295          LD    A,CR
'01FF  CD1601'   0296          CALL  SEROUT
'0202  CD15E0    0297 BINPLP   CALL  QCKCHK
'0205  2019      0298          JR    NZ,INDONE-$
'0207  DBFD      0299          IN    A,(0FDH)
'0209  CB4F      0300          BIT   1,A
'020B  28F5      0301          JR    Z,BINPLP-$
'020D  DBFC      0302          IN    A,(0FCH)
'020F  CBBF      0303          RES   7,A
'0211  FE00      0304          CP    MEOF;NOT THIS!
'0213  28ED      0305          JR    Z,BINPLP-$
'0215  FE0A      0306          CP    LF        ;NOT WANTED:
'0217  2800      0307          JR    Z,INMEM-$;SKIP-$
'0219  12        0308 INMEM    LD    (DE),A
'021A  13        0309          INC   DE
'021B  CD7002'   0310 SKIP     CALL  FSEND
'021E  18E2      0311          JR    BINPLP-$
'0220  3E0D      0312 INDONE   LD    A,CR
'0222  12        0313          LD    (DE),A
'0223  13        0314          INC   DE
'0224  3E0A      0315          LD    A,LF      ;IF
'0226  12        0316          LD    (DE),A    ;<LF>
'0227  13        0317          INC   DE        ;NEEDED
'0228  3E00      0318          LD    A,MEOF
'022A  12        0319          LD    (DE),A
'022B  13        0320          INC   DE
'022C  C3BE01'   0321          JP    ADONE
                 0322 ;THE FOLLOWING IS USED IN BOUT
                 0323 ;INSTEAD OF INPIFR TO PERMIT
                 0324 ;COMPUTER TO CONTROL TERMINAL
                 0325 ;RATE OF TRANSMISSION
'022F  F5        0326 BINIFR   PUSH  AF
'0230  DBFD      0327          IN    A,(0FDH)
'0232  CB4F      0328          BIT   1,A
'0234  2002      0329          JR    NZ,BSHOW-$
'0236  F1        0330 BOUTRT   POP   AF
'0237  C9        0331          RET
'0238  DBFC      0332 BSHOW    IN    A,(0FCH)
'023A  CBBF      0333          RES   7,A
'023C  CD7002'   0334          CALL  FSEND
'023F  F1        0335          POP   AF
'0240  C9        0336          RET
'0241  F5        0337 BSOUT    PUSH  AF
'0242  DBFD      0338 BWBE     IN    A,(0FDH)
'0244  CB47      0339          BIT   0,A
'0246  CD2F02'   0340          CALL  BINIFR
'0249  28F7      0341          JR    Z,BWBE-$
'024B  F1        0342          POP   AF
'024C  A7        0343          AND   A
'024D  EA5202'   0344          JP    PE,BOUTOK
'0250  CBFF      0345          SET   7,A
'0252  D3FC      0346 BOUTOK   OUT   (0FCH),A
                 0347 ;LINE HANDSHAKE (AF DESTROYED)
                 0348 ;
'0254  CBBF      0349          RES   7,A
'0256  FE0D      0350          CP    CR
'0258  C0        0351          RET   NZ
'0259  CD15E0    0352 LWAIT    CALL  QCKCHK
'025C  C0        0353          RET   NZ
'025D  DBFD      0354          IN    A,(0FDH)
'025F  CB4F      0355          BIT   1,A
'0261  28F6      0356          JR    Z,LWAIT-$
'0263  DBFC      0357          IN    A,(0FCH)
'0265  CBBF      0358          RES   7,A
'0267  CD7002'   0359          CALL  FSEND
'026A  FE0A      0360          CP    LF
'026C  C8        0361          RET   Z
'026D  18EA      0362          JR    LWAIT-$
'026F  C9        0363          RET
'0270  FE20      0364 FSEND    CP    ' '
'0272  D20CE0    0365          JP    NC,SEND
'0275  FE0D      0366          CP    CR
'0277  CA0CE0    0367          JP    Z,SEND
'027A  FE0A      0368          CP    LF
'027C  CA0CE0    0369          JP    Z,SEND
'027F  FE00      0370          CP    0
'0281  C8        0371          RET   Z
'0282  FE07      0372          CP    07H
'0284  2816      0373          JR    Z,BELL-$
                 0374 ;MUST BE SOME OTHER
                 0375 ;CONTROL CHARACTER
'0286  F5        0376          PUSH AF
'0287  CDEDE1    0377          CALL  0E1EDH
'028A  3E3D      0378          LD    A,'='
'028C  0604      0379          LD    B,4
'028E  CD0CE0    0380 SMR      CALL  SEND
'0291  10FB      0381          DJNZ  SMR-$
'0293  F1        0382          POP   AF
'0294  C9        0383          RET
'>0295           0384 BELMSG
'0295  3C        0385  DEFB '<'
'0296  42        0386  DEFB 'B'
'0297  45        0387  DEFB 'E'
'0298  4C        0388  DEFB 'L'
'0299  4C        0389  DEFB 'L'
'029A  3E        0390  DEFB '>'
'029B  00        0391  DEFB 00
'029C  E5        0392 BELL     PUSH  HL
'029D  219502'   0393          LD    HL,BELMSG
'02A0  CDBAE1    0394          CALL  SNDLIN
'02A3  E1        0395          POP   HL
'02A4  C9        0396          RET
                 0397 ;*********END*****************
ERRORS=0000
ABIT1     00C9 ADONE         01BE BELL      029C
BELMSG    0295 BINIFR        022F BINP      01F8
BINPLP    0202 BITEND        00D4 BLOOP     0038
BOUT      01CD BOUTLP        01D2 BOUTOK    0252
BOUTRT    0236 BSHOW         0238 BSOUT     0241
BVWLP     012F BWBE          0242 CCEOF     001A
CODED     0070 CONTBL        ECBE CR        000D
CRLF      E205 DEBOUN        0042 DONE      01B0
DONELP    01B4 DUPLEX [INT]  0000 FINEND    00F7
FINOP     00B7 FSEND         0270 FTOKEN    E225
GETADR    01C6 GETIY         E1A2 GRATBL    EC6E
GTOKN2    E22F INDONE        0220 INMEM     0219
INPIFR    0104 INSTBL        EC1E KEYBD2    0003
KEYRET    0102 LF            000A LFDFT     01E6
LSTKEY    006C LWAIT         0259 MCONVA    E23D
MEOF      0000 MINPUT        E13A MLOOP     002B
MUNCNV    E1E8 NOGRAP        0095 NONCON    0062
NONGRA    005C NONSHI        0068 NORPT     0025
OUTDLY    FF01 OUTEND        01F1 OUTOK     0127
PLP       0187 QCKCHK        E015 QQ        01A8
REPET     001A SDLYLP        01DC SECEND    00E6
SEND      E00C SEROUT        0116 SHITBL    ED0E
SHOW      010D SKIP          021B SKIP1     009F
SKIP2     00A8 SKIP3         00B1 SKIP4     00B4
SKIP5     00D2 SKIP6         00F0 SLOOP     0034
SLOTBL    ED5E SMARTS        016F SMR       028E
SNDLIN    E1BA TERMNL        0156 UNSTBL    EDAE
VCRLF     0151 VDLY          00FF VDLYLP    013B
VIEW      012A WAITK         00C1 WBE       0117
```

signals (the DEC-20 ignores this bit, anyway). By recoding lines 198 and 199, you can have odd parity, mark parity or space parity.

There are many choices to consider associated with the block transfers. In addition to deciding what to do about the parity bit for transmission and for storage in RAM, you should decide whether lines are to be terminated with just <CR> or with both <CR> and <LF>.

There are several places where this decision will affect the code. For example, lines 306 and 307 as coded are not needed (I do need the <LF> in RAM), but if you do not want the <LF> in RAM, the relative destination in line 307 should be changed to "SKIP-$." There are several places where you may wish to change delay constants, such as VDLY (which is used in VIEW). You may want to also change MEOF and/or CCEOF.

You may have different ideas on how to use the e(X)it command. I have coded it as a jump to the cold-entry point of the monitor (see line 256). This will disable the RS-232 port and reestablish the cassette interface; the baud rate will be changed back to 1200. You may wish to jump to the warm-reentry point of the monitor at 0E003H or to write some more software and jump to that.

I have used this program to work on several projects in addition to developing this article, and I'm sure those of you who try this program will enjoy it. ■


How's your love life?
INTERLUDE #99 THE ULTIMATE EXPERIENCE...
A little dull around the edges? Routine? Predictable? Boring? Maybe all it needs is a little Interlude. Interlude is the most stimulating computer game ever conceived. It combines a computer interview, an innovative programming concept, and a one-of-a-kind manual to turn your love life into exciting, adventurous, delicious fun!
Interlude is: romantic . . . playful . . . outrageous . . . a fantasy. Interlude is: ■ Wet fun on a hot summer night. (Interlude #21) ■ A surprise on the way home from dinner. (Interlude #42) ■ A bubble bath that ends with a bang. (Interlude #78) ■ An evening to rest while she does all the "work." (Interlude #25) ■ The most romantic of evenings. (Interlude #84) ■ A new twist to an old subject. (Interlude #69) ■ Just watching her . . . (Interlude #57) ■ An erotic fantasy! (Interlude #33)
With over 100 Interludes, you can satisfy all levels of interest and desire. Each Interlude is fully described in the manual, and the more elaborate ones are detailed with regard to settings, props, and mood-enhancing techniques. But we've saved a few super Interludes for that very special time when your interview indicates you're ready! At that time, you will be introduced to one of several Interludes held secret within the computer. (When you learn secret Interlude #99, your love life may never be the same again!) Interlude can give you experiences you'll never forget. Are you ready for it?
Interlude™
The Ultimate Experience.
Interlude, 10428 Westpark, Houston, Texas 77042 I'm really ready! Rush me ____ copies of Interlude today.
235
☐ For the Apple II (16K) # ☐ For the TRS-80 (Level II-16K) ## ☐ $14.95 for cassette ☐ $17.95 for diskette.
Add $1.50 for shipping. Texas residents add 6% sales tax. My check (payable to Interlude) is enclosed. ☐
*Charge my ☐ MASTERCHARGE ☐ VISA account.
Account No. ____ Expiration date ____
All charge customers must sign. Signature ____
Name ____ Age ____
Address ____
City ____ State ____ Zip ____
*CHARGE CUSTOMERS: Order by phone toll-free! 1-800-327-9009 Ext. 306 ,Florida-1-800-432-7999 Ext. 306
Apple II is a registered trademark of Apple Computers, Inc. ## TRS-80 is a registered trademark of Radio Shack, a Tandy Co.

A Hexadecimal Front Panel for Z-80 Systems

A 4 MHz processor board from Ithaca Intersystems forms the core of this construction.

John D. Ciana
24 Eichstrasse
8045 Zurich
Switzerland

This article describes my search for, and utilization of, a processor board. I was looking for an inexpensive Z-80 processor, with all the usual gadgets; but most of all, it had to have a monitor program in ROM.

Unfortunately, none of the boards I looked at met all of the above conditions. Most provided a ROM socket, but no chip. ROM chips that were provided with the boards were usually as dull as when they came out of the production line. The boards were either too expensive, or the monitors resided in low memory.

The one that came the closest was the Morrow front panel computer board, which is "only" an 8080, and the keyboard and display are in octal. So, I asked myself, "Why a monitor?" With a versatile front panel and a good floppy-disk operating system, you don't need a block move or a memory clear function. So, the question is: What are the desirable features of a modern front panel for a (nearly standard) S-100 computer?

First, it is easy to read the data and address values on the display. Thus, the LEDs are excluded in favor of the hex display to show the state of the buses. Second, there should be a way to enter data without having to toggle all those switches (see Photo 1). Four easy strokes on a hex-encoded keyboard are required, instead of the error-prone 16 switches to flip up or down. Third, the single-step feature allows you to stop the program and step slowly from then on.

A novel feature of this front panel is the hardware breakpoint. Just by putting an address on the keyboard display and flipping the breakpoint switch, you can set a trap that will stop the processor right on that address, if it ever executes through it. This allows the programmer to halt the 'brain' just at the right place, read the memory and then continue processing.

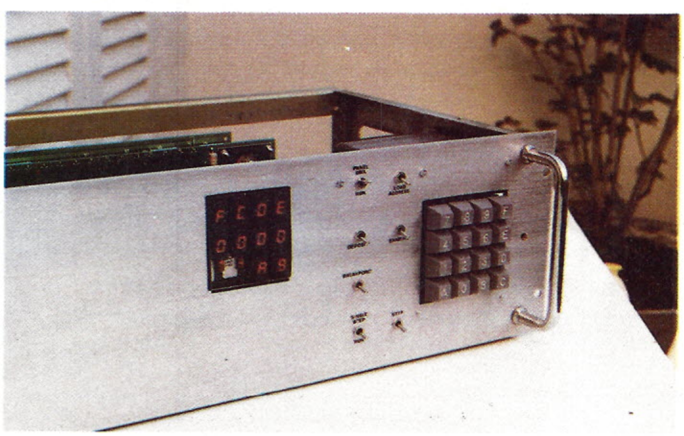


Photo 1. Hex-encoded keyboard.

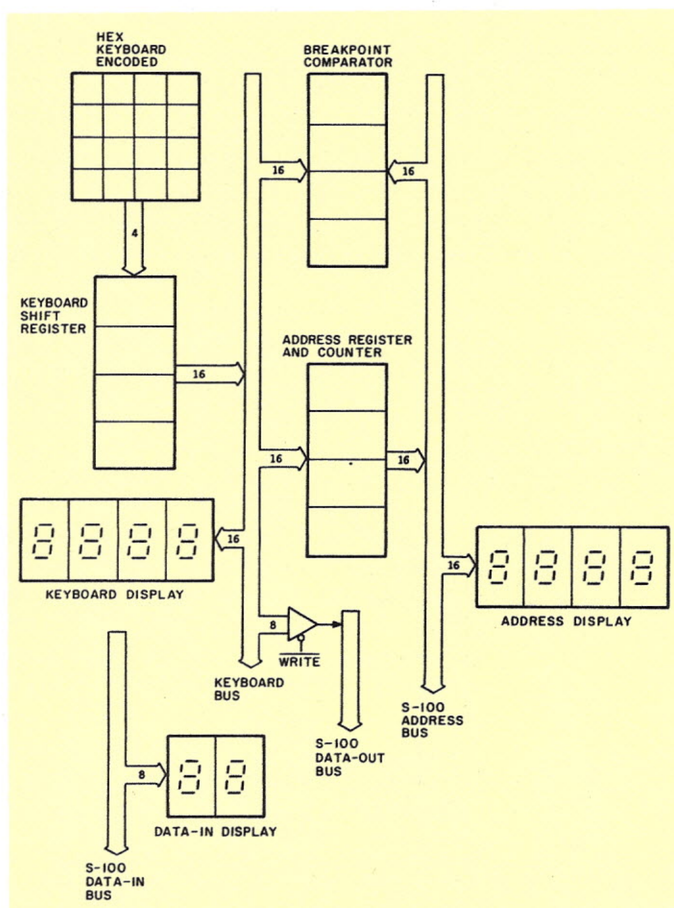


Fig. 1. Front panel overview.

Thus, I decided to build a front panel and to buy an off-the-shelf processor board. I chose a 4 MHz Z-80 processor board from Ithaca Intersystems. It has worked fine since I first powered it up.

What's New

Technology grows so fast these days that by the time you can get the manufacturer's manual on your desk—and we are not left behind here in Europe—the largest OEMs have had their advance copy of spec sheets for six months. Thus, as an engineer or hobbyist, you are at least six months behind the state of the art.

I am thinking about using complementary MOS integrated circuits, so the best thing to do is obtain the specification manual of one or two of the biggest MOS chip vendors and use the bonanza of low-cost chips on the market. The highly sophisticated circuits allow you to reduce the package count and to build functions not possible before.

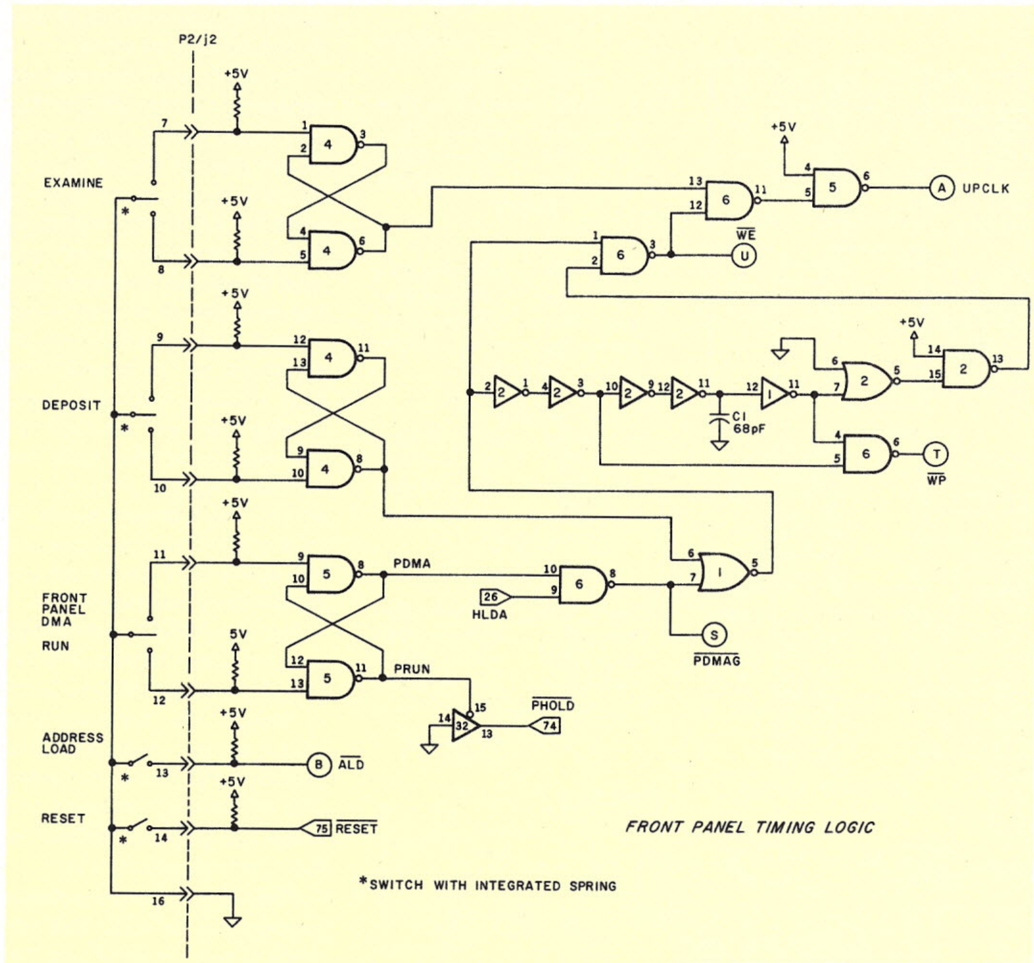
The added advantage is that you can mix existing TTL functions and CMOS by using the LSTTL Schottky family. You must remember that at maximum, two LSTTL loads connect to one CMOS output; don't use the usual TTL because the output high voltage of 3.5 V is too low to obtain sufficient noise margin when connected to a CMOS input. CMOS technology is five to ten times slower than Schottky TTL, but this is no disadvantage, as we shall see later.

The workhorse of this front panel is the MC14495 latched hex decoder and display driver. No current limiting resistors are needed, because this is taken care of on the chip itself. The MC14572 multigate IC allows an optimal mix of four inverters, one NAND and one NOR on one chip. The added benefit of using CMOS and LSTTL together is much lower power consumption. Thus, only one 5 V regulator is needed on this board.

Surroundings

My front panel board is a "nearly S-100" accessory. I am using a fully static memory board with 300 ns, 2114 memory chips. When the front panel wants the bus, the processor releases it, so there is no fighting for signal lines. The front panel uses DMA techniques, providing all signals necessary to my memory and allowing even the processor board to be removed without affecting Deposit or Examine functions.

Thus, you can't use the refresh counter built in the Z-80 because once the front panel grabs the bus, the processor is Tri-stated away from the bus. If you plan to use dynamic memories, make sure that they use on-board refreshing and that the bus "sees"



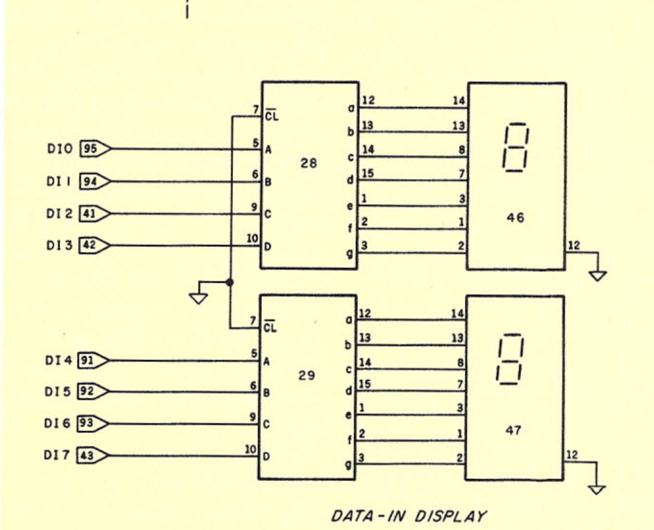


Fig. 2a. Front panel timing logic and Data-in display.

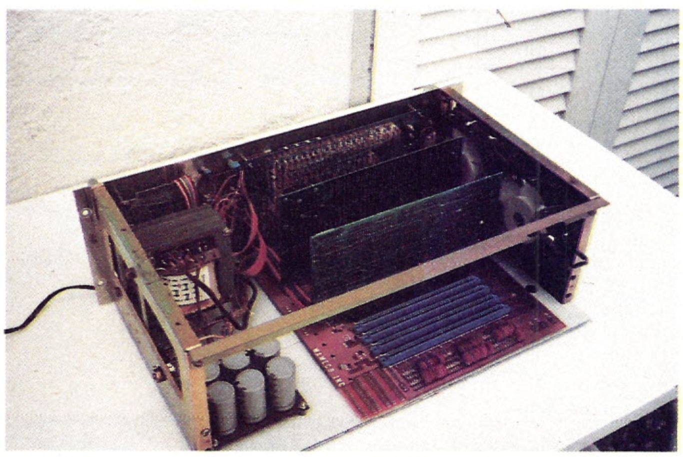


Photo 2.

a static-like memory.

Functional Description

The hexadecimal display contains three lines of 7-segment displays and eight status LEDs in the lower left corner of the display window. The upper line of displays shows the state of the 16-bit keyboard register on its four digits. Each digit represents four bits and is decoded at each of its 16 different states.

The second line represents the 16-bit S-100 Address bus, and the two lower digits represent the 8-bit S-100 Data-in bus. The keyboard register feeds a Keyboard bus that is not part of the S-100 system but is extensively used throughout the board (see Fig. 1). While writing in memory with the Deposit switch, the eight lower bits of the Keyboard bus will be sent out on the Data-out bus (the two rightmost digits on the keyboard display).

The 16-bit address register and counter will be loaded from the 16-bit keyboard register with the Load Address switch and will count up at each of the Examine or Deposit switch strokes, thus stepping the memory sequentially. Fig. 1 shows the 16-bit breakpoint comparator that gives an equal condition to the breakpoint logic when the Keyboard bus and the Address bus are the same—at SM1 processor time. This permits you to enter the panel DMA mode and examine and respectively write the memory. By leaving the breakpoint latched state, processing will be resumed.

Figs. 2a, 2b and 2c show more details. Every letter in a circle connects to another figure, while each number in a flag shows an S-100 bus connection. Fig. 2a shows that the S-100 Data-in bus is decoded and displayed constantly thanks to ICs 28 and 29 and displays 46 and 47.

The front panel switches are connected through P2/J2 to the front panel timing logic. I used ground level in the wiring and pull-up resistors in the logic to avoid the +5 V in the wiring, so that any short-circuit will not have disastrous consequences. The reset switch is directly fed to the S-100 bus, having straight action on the processor. The ALD, address load line, when pulsing low, will parallel-load the address counter from the keyboard register (Fig. 2c, ICs 16–19). The address counter is Tri-stated on the bus only when the drivers (ICs 30–33) are enabled during front panel DMA.

The term $\overline{\text{PDMAG}}$, (panel DMA granted) is generated on Fig. 2a. Once the front panel DMA switch is flipped, the corresponding latch is set, requesting $\overline{\text{PHOLD}}$ to the processor. This one answers with HLDA, thus releasing the buses and granting DMA to the front panel at gate 6, pin 9.

$\overline{\text{PDMAG}}$ enables the address counters on the bus. This address bus is decoded and displayed all the time by ICs 24–27 and 38–41, respectively. The same term also enables Tri-state driver 36, allowing some necessary signals on the bus. The processor Tri-state buffers are disabled by $\overline{\text{STADSB}}$, $\overline{\text{ADDDSB}}$, $\overline{\text{DODSB}}$ and $\overline{\text{CCDSB}}$ going low. PSYNC, SOUT and SINP are inactive low in this design. PDBIN and SMEMR are high during a memory read and low during a write, according to $\overline{\text{WE}}$, write enable.

Inverting buffer 37 will drive six LEDs to show the six statuses, SOUT, SINP, SM1, MRQ, PWAIT and SMEMR. You could display other statuses or add

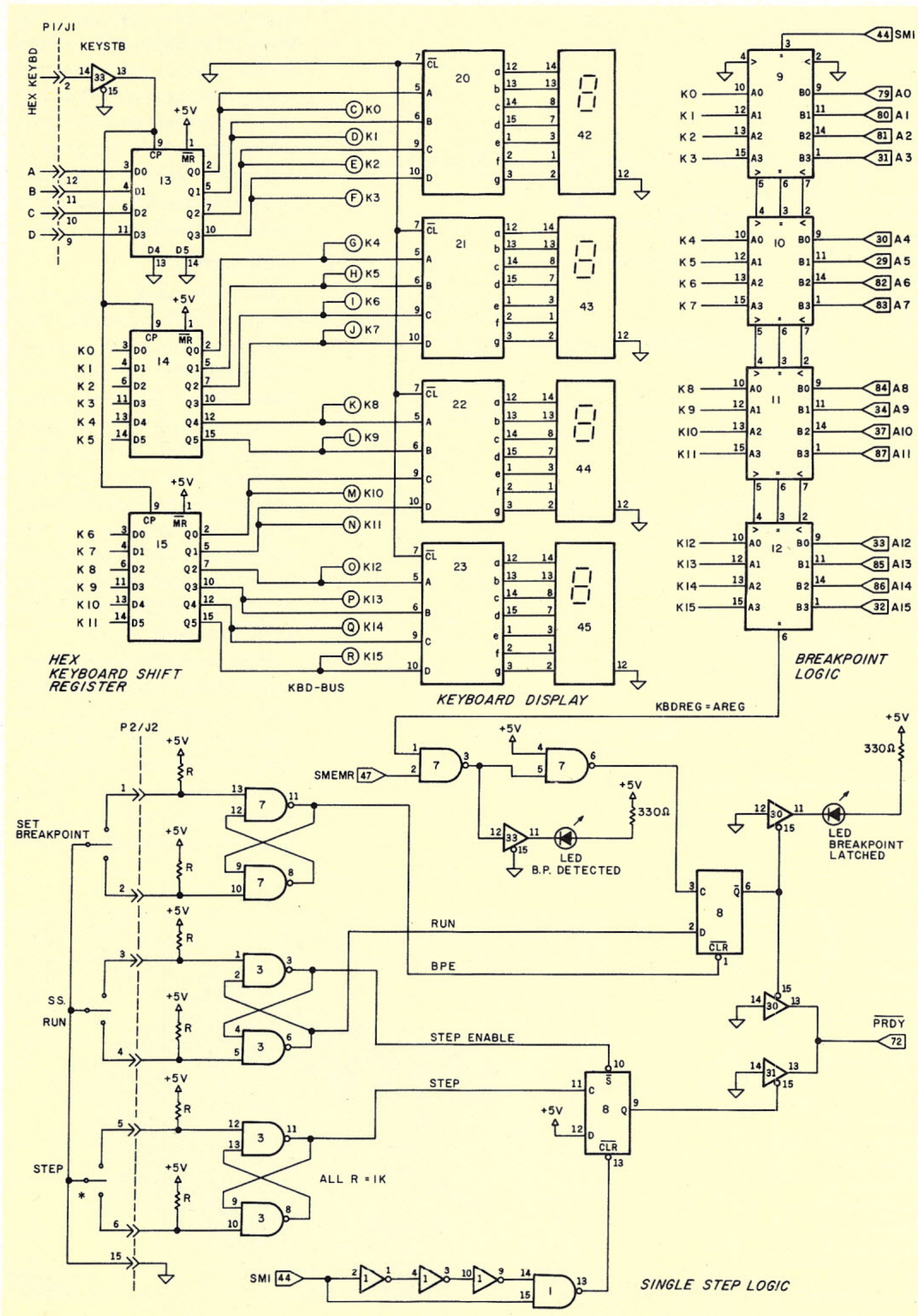


Fig. 2b. Keyboard register circuitry.

Most small system users think all microcomputers are created equal. And they're right. If you want performance, convenience, styling, high technology and reliability (and who doesn't?) your micro usually has a price tag that looks more like a mini. It seems big performance always means big bucks. But not so with the SuperBrain.

Standard SuperBrain features include: twin double-density 5¼" drives which boast over 300,000 bytes of disk storage. A full 32K of dynamic RAM - easily expandable to 64K. A CP/M* Disk Operating System which insures compatibility to literally hundreds of application packages presently available. And, a 12" non-glare, 24 line by 80 column screen.

You'll also get a full ASCII keyboard with an 18 key numeric pad and individual cursor control keys. Twin RS232C serial ports for fast and easy connection to a modem or printer. Dual Z80 processors which operate at 4 MHZ to insure lightning-fast program execution. And the list goes on. Feature after feature after feature.

Better yet, the SuperBrain boasts modular design to make servicing a snap. A common screwdriver is about the only service tool you'll ever need. And with the money you'll save on purchasing and maintaining the SuperBrain, you could almost buy another one. For under $3,000, it is truly one of the most remarkable microcomputers available anywhere.

Whether your application is small business, scientific or educational, the SuperBrain is certainly one of today's most exciting solutions to your microcomputer problems. Call or write us now for full details on how you can get big system performance without having to spend big bucks. So, why not see your local dealer and try one out today. Intertec systems are distributed worldwide and may be available in your area now.

INTERTEC DATA SYSTEMS®

2300 Broad River Rd. Columbia, SC 29210
(803) 798-9100 TWX: 810-666-2115

*Registered trademark of Digital Research, Inc.

SUPERBRAIN™

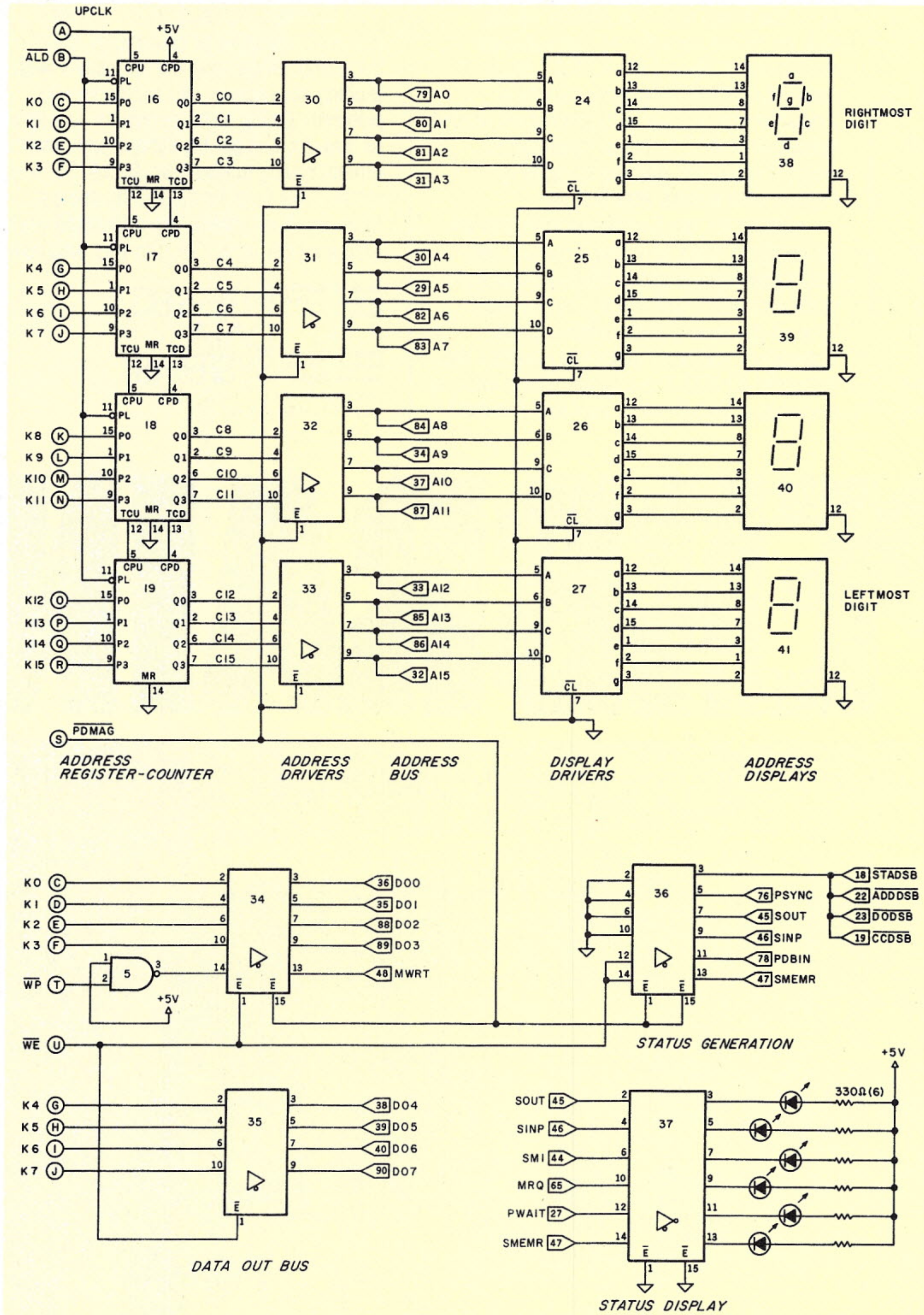


Fig. 2c. Address bus.

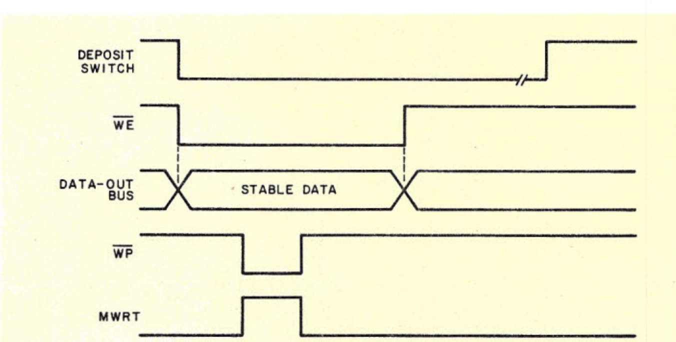


Fig. 3. Write timing.

more LEDs and drivers. On Fig. 2c, the Data-out bus is driven by the eight lower bits of the keyboard register, but only when write enable goes low through Tri-state drivers 34 and 35.

The write clock to memory comes either from the processor (MWRT) or from the front panel ($\overline{\text{WP}}$) write pulse. For this, the second half of driver 34 is used, with the enable pin 15 tied to $\overline{\text{PDMAG}}$. Make sure that MWRT goes Tri-state at the processor during panel DMA.

When not writing, the panel reads memory all the time at the address given by the address register-counter. By toggling the Examine switch, you merely step the counter up by one, through the UPCLK term. Writing in memory is more complex and requires a sequence of signals (see also Fig. 3).

In Fig. 2a, as soon as the Deposit switch is toggled, $\overline{\text{WE}}$ goes low, enabling the Data-out bus drivers. $\overline{\text{WE}}$ is generated on IC6, pin 3, as long as the output signal of IC 1, pin 5, does not yet reach the end of the inverter chain made by the gates of ICs 1 and 2. CMOS gates are slow, and, with the adjunction of C1, I expect $\overline{\text{WE}}$ to be around 500 ns long. Since the generation of $\overline{\text{WP}}$ uses three of the same gates used by $\overline{\text{WE}}$, I expect $\overline{\text{WP}}$ to go low about 120 ns after $\overline{\text{WE}}$ does and remain low for about 250 ns.

The write pulse occurs when the Data-out bus is long stabilized and stops long before the bus is released. With this timing, I never had any problems writing in memory. You may have to adjust the value of C1, going as low as 20 pF or even removing it, to as high as 100 pF. When $\overline{\text{WE}}$ goes high, the address counter is toggled to the next memory byte by UPCLK.

The keyboard register is shown in Fig. 2b. It is made of three shift register chips, each having six independent bits. It is actually wired in four 4-bit words placed one after the other. I use a commercial keyboard made locally. The hex keyboard by Poly-Packs should be fine for this project.

For those of you who want to build one yourself, mine uses a National MM74C922 encoder chip. The four data bits are true high at connector P1/J1, and the keyboard strobe KEYSTB should pulse low. The decoder-drivers 20–23 and the displays 42–45 show the keyboard bus all the time, and each typed digit will appear on the right and move to the left until it disappears off the display.

The breakpoint logic is made with the comparator ICs 9–12.

When the Keyboard bus and the S-100 Address bus are the same, only the SM1 signal is needed on the equal input on IC9, pin 3. The equal condition will ripple through the comparators and will reach a gate in IC7. This signal NANDed with SMEMR (memory read) will light the "breakpoint detected" LED and will clock the breakpoint flip-flop.

This clocking will only be effective if the clear is removed on the flip-flop, pin 1. The term BPE should be high, that is, the Breakpoint switch flipped on. The setting of the flip-flop will light the "Breakpoint latched" LED and set the PRDY line low to the processor, halting it right at SM1 time.

To leave this situation, flip the switch back to the off position, clearing the flip-flop and releasing the processor. This logic is disabled during single step because it has no use as long as the RUN term is low at the D input of the flip-flop.

The single-step logic is made of the second half of IC8. When the single-step/run switch is on RUN, the STEP ENABLE line is low, setting the flip-flop all the time, albeit clearing pulses coming from SM1. The Q output remains high, disabling the driver in IC31 and leaving the processor alone. In the single-step position, the enable line goes high, and the next SM1 signal from the processor clears the flip-flop.

The pulse generator in IC1 should now seem familiar. The Q output goes low and sends $\overline{\text{PRDY}}$ to the processor, stopping it still at SM1 time. To advance one instruction byte, the Step switch will clock the flip-flop, releasing the processor, but just enough to go to the next SM1 time, thus realizing the single-step effect.

Construction

Use the flexible wire-wrap construction techniques for this one-time front-panel project. I chose a Vector 8800 V wire-wrap board (see Photo 3). It is important to use a board with vertical PC runs, so that you can place the 7-segment display vertically. The layout is not critical (see Fig. 4 or design your own). Remember to wrap systematically —bus after bus, registers, counters, and miscellaneous logic at the end. In this way, you will avoid forgetting a wire (see Photo 4). Use a liberal amount of decoupling capacitors (every three ICs, for example) on the +5 V bus (0.01 to 0.1 uF).

I bought cut and stripped wire for the shortest lengths, because these are the most difficult to manipulate. It took me six hours to "wrap it up," and I didn't forget a wire. It worked immediately! The places marked R on Fig. 4 are 16-pin wire-wrap sockets, containing 14 resistors. I bought these in flat strips of coated ceramic, containing seven resistors each and having one common pin. Each socket gets two strips.

Tests

Wire-wrap the board, solder the capacitors and wire and solder the +5 V regulator. Unplug all boards from the S-100 bus and plug-in your new board. Power-up. Measure the voltage on the +5 V bus. If the test is positive, plug all ICs and displays, resistors and flat cables to the switches and keyboard, respecting their orientation. Power-up. The display should light up.

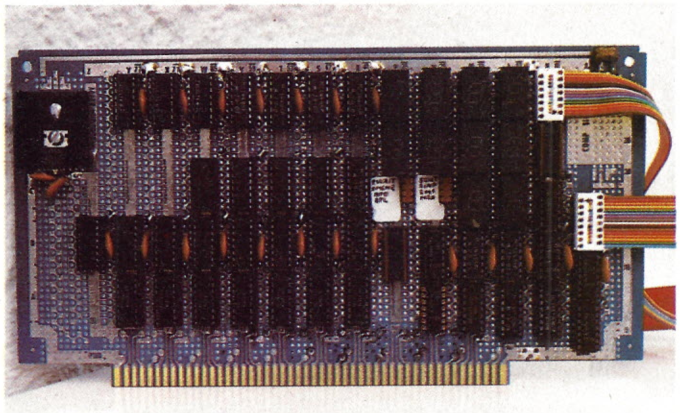


Photo 3. PC board design.

Toggle the Examine and Deposit switches when on panel DMA. The address display should count up. Punch a few figures on the keyboard, which should appear on the right and shift to the left of the keyboard display. Power-down. Plug in the memory board. Without the processor, you should be able to read and write to memory. Plug the processor in the S-100 bus.

Now you can try the test program. If there are malfunctions, try to determine if "fighting" for a line or a set of lines on the bus exists. When in panel DMA, the processor should release the bus. Remember that the Address and Data-in displays show bus data all the time, enabling you to troubleshoot. This program will load the 16-bit BC register from a pair of bytes in memory and store it back at another place. See Table 2.

1. Flip the switch to panel

IC Number	Description
1,2	MC14572 CMOS multigate chip (Motorola)
3,4,5,6,7	74LS00 TTL four 2-input NAND
8	74LS74 TTL double D-flip-flop
9-12	74LS85 TTL 4-bit comparator
13,14,15	74LS174 TTL 6-bit shift register
16-19	74LS193 TTL 4-bit up-down counter
20-29	MC14495 CMOS Hex decoder and display driver (Motorola)
30-36	74LS367 TTL Tri-state hex buffers, non-inverting
37	74LS368 TTL Tri-state hex buffers, inverting
38-47	DL 702 LED 7-segment displays, common cathode

Table 1. Integrated circuits.

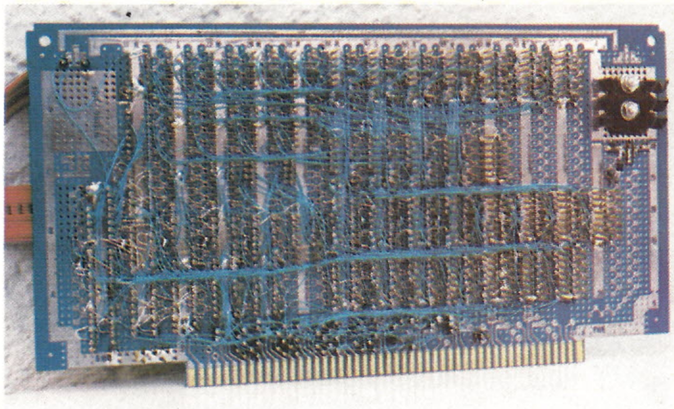


Photo 4. Wire-wrap construction.

At address 0000:

ED 4B 00 02	Loads contents of addresses 0200 and 0201 in register pair BC.
ED 43 02 02	Stores contents of register BC in memory at address 0202 and 0203
C3 00 00	Jumps back to 0000, beginning of program

Table 2. Test program.

DMA.

2. Load 0000 in the keyboard display and toggle Load Address.

3. Type in ED and toggle the Deposit switch.

4. Type in 4B and toggle the Deposit switch.

5. Enter 00, 02, ED, 43, 02, 02, C3, 00, 00, using the Deposit switch each time, as above.

6. Load 0200 in the keyboard display and toggle Load Address.

7. Deposit 55, CC, 00, 00.

8. Switch single-step on and panel DMA off.

9. Hit Reset. (The processor will begin to execute at 0000.)

10. Toggle-step a few times until you see C3 on the Data-in bus. You will then have executed both the load and the store.

11. Switch panel DMA on and single-step off.

12. Check at address 0200. You should get 55, CC, 55 CC, proving that the processor moved data.

13. Switch panel DMA off, releasing the full power of your Z-80. It should loop forever in this small program, until you try point 14.

14. Load 0000 in the keyboard register and flip the Breakpoint switch. It should stop at address 0000, showing ED on the Data-in bus. Both the breakpoint detected and breakpoint latched LEDs should come on. Flip the switch off, and the processor will be released. Try this with other addresses.

15. Write other programs and have fun.

16. Write articles.

Conclusion

This article shows a way to speed up program development, using minimum hardware and no support software. It should work with any S-100 system, but has only been tested with the mentioned hardware. It should be easy to boot a disk system by just typing a few digits on the keyboard. If you know where to buy the chips, it should not be expensive. ■

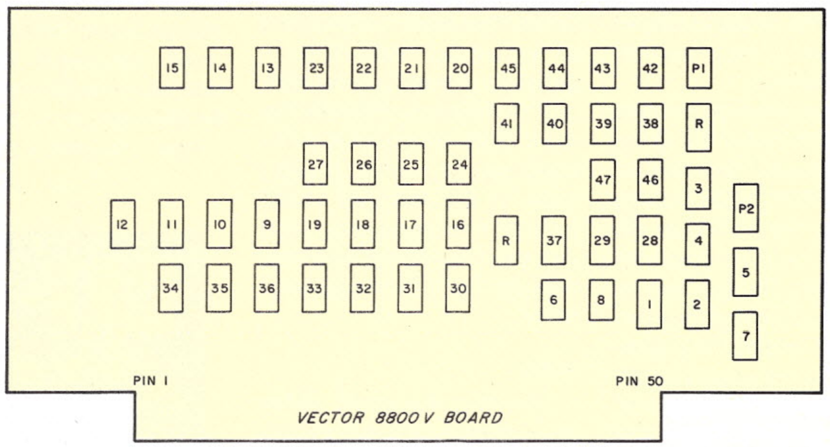
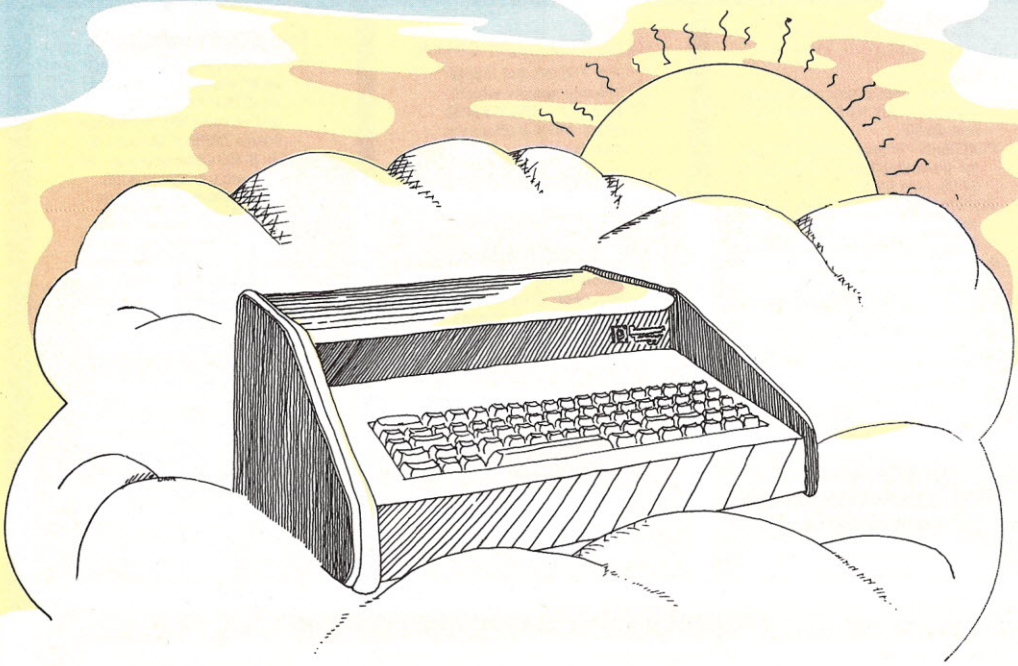


Fig. 4. Front panel board, component side.

The Sol Also Sets



B. R. Evans
16 Marwin Rd.
Pickering, Ontario
Canada L1V 2N7

As the business side of microcomputing becomes bigger and more competitive, and as the giants take over a larger share of the market, it is inevitable that some of the smaller companies will fail. It is just as inevitable that some of you reading this article will soon possess "orphans." It is to you that I address this article.

I own a Sol-20, an 8080A-based microcomputer that I considered one of the best on the market. Unfortunately, it is no longer on the market! One of my first thoughts on hearing of the demise of Processor Technology Corp., the Sol's creator, was to try to unload it on somebody. However, after some clear thinking, it became obvious that the same features that had convinced me to buy this machine two years ago were still there.

I then began a methodical process to cushion the blow and to protect my investment in the future. It is about this that I am writing, and I hope that some of you reading my article will benefit from my work.

Safety in Numbers

Before buying a microcomputer, you should join its users group. Get as many back issues of their newsletters as possible. These groups have a wealth of knowledge of software bugs and hardware quirks. If Processor Tech had had this much talent on tap, it never would have failed. Users groups can also aid you in evaluating which equipment is compatible with your system. Share your experiences—both your successes and failures—with them.

If a users group is sufficiently large, it can make an offer to a defunct company to buy some of the source listings of the software and distribute them to its members. You will appreciate this when you have to modify your software to interact with that of companies that didn't even exist when your machine was in production.

Saving for a Rainy Day

The second step is to obtain as much of the documentation—schematics, change notices and source listings—for your equipment as possible. Many of the computer stores that previously carried your brand will be happy to dispose of these and may not even charge you for them. Some computer stores, such as the Computer Port in Arlington, Texas, will sell documentation packages such as these.

Next, you must keep track of the service facilities. Computer stores have a terrible turnover rate, exceeded only by the turnover of technicians. If you know of a technician who seems to know everything about your machine, cultivate his friendship, check to see that he is still on staff at the same store and patronize the new store if he moves to a new one or opens his own. Remember that many of the technicians you may approach in the future will have no idea of the flukes that are present in your computer.

If you have been planning to add to your system "sometime in the future," now is that time. Software compatibility with your cassette format will quickly dry up. Do you really want to enter an 8K program in BASIC through the keyboard? Modifications that some companies have made to their peripherals, such as floppy disks, will soon disappear with the market. However, don't be stampeded into buying peripherals that use standard outputs such as serial or parallel ports. You should have no difficulty writing your own drivers for them at any time.

Keep your eyes open for companies selling off equipment made by the defunct company. If the equipment was sound before "the fall," it still will be and will cost less. Be prepared to dicker with the owner of the computer shop. He wants to get rid of the merchandise as soon as possible.

Conclusion

The loss of a small computer company is never a propitious event. We all lose. The owner of the company's products is inconvenienced, and we all are deprived of the ingenuity that is the trademark of the small electronics firm. However, I hope I have presented some ideas that will help you to survive the setting of your own personal Sol. ■

Reader-Service Reply: A Record-Keeping System

This program is helpful for making judicious use of reader-service cards.

Howard J. Siegel
15020 North 7th Drive
Phoenix, AZ 85023

I use the reader-service reply cards in *Kilobaud Microcomputing* to request information on products and services of advertisers. Wayne encourages us to use these cards because they help to keep readers informed to make intelligent decisions about microcomputer equipment/software purchases. Advertisers can measure the effectiveness of their ads. That helps to keep current advertisers and obtain new ones, providing you and me with an expanded magazine (both in articles and advertising) and the opportunity to circle new reader-service numbers.

To become more informed and to expand the size of the magazine, I remove the card from the back of each issue and keep it with me as I read, looking for new and different ads to make requests from.

However, I read many magazines. Each has its reader-service reply card. Many times I have received duplicate data from the same company because I forgot a previous request from an ad in another magazine. So what? Well, I cost the company needless expenditure. That extra cost may be built into the next piece of equipment or software package I want to buy.

I wrote the accompanying program to help prevent this from happening. This program allows me to keep track of the companies to whom I have made requests. The basic functions performed are adding, deleting and checking on the information stored.

System Requirements

My system is a Polymorphic 8810 with 32K of RAM and two disk drives, although disks are not required. The program requires about 3.5K of memory and is written in Poly disk BASIC version B08C.

The following are some of the notations that may be unique to Poly BASIC:

DIM L$(1:50)—this specifies a string array consisting of one member with a maximum length of 50 characters.

MID$(L$,I,I)—function that returns the Ith character of the string. MID$(L$,I,J) returns the Ith through the Jth characters of the string, inclusively.

ASC(string variable)—function that returns the decimal representation of the ASCII code for the first character in the specified string.

VAL(string variable)—function that returns the numeric value of a numeric string if the string doesn't contain blanks.

CHR$(expression)—function that returns a string specified by the expression. This expression is a decimal representation of the ASCII code. CHR$(12) causes a skip to top of page on a printer or clear screen and home up on a CRT. CHR$(13) is the equivalent of a carriage return.

ON ERROR—provides the user with control over error-recovery procedures. Use of this statement is explained below.

Program Notes

When I wrote this program, I had not added disk to my system. In order to maintain the information on which companies I had requested literature from, I stored the company name, magazine name and issue date in DATA statements. Poly BASIC provides two features that help to dynamically create and update these DATA statements.

The system has a 64-character "type-ahead" buffer, which allows the operator to manually input data before the program asks for it or to create that input stream as a part of the program itself.

The second Poly feature is the ability of the user to invoke, from BASIC, an 8080 OUT instruction to a specified port. Port 0 is the type-ahead buffer. The instructions in Example 1 enter each character in the string L$ into the type-ahead buffer. If the program branches to STOP, the characters in the buffer are read, and the BASIC program line number 10 is either created with or replaced by the contents of the buffer.

```
1000 L$="10DATA XYZ COMPANY—KB 6-79"
1010 FOR I=1 TO LEN(L$)
1020 A$=MID$(L$,I,I)
1030 OUT0,ASC(A$)
1040 NEXT I
```

Example 1.

If the last string put into the buffer is run, followed by a carriage return (CHR$(13) in Poly BASIC), the program automatically restarts with the new information in the proper DATA statement. The DATA statements for the literature requested are built into lines at the start of the program. They are built and output to the buffer by the function FNL(L$) starting in line 1400.

Program Functions

The first is the ADD function. The program simply asks for the input of the data to be saved. I enter company name, magazine requested from and issue date. To keep the output looking neat, with the request for input I put an "I" on the screen to indicate the maximum size allowed for the company name. The program then builds a string using the next available BASIC statement number and outputs it to the buffer.

Another string is created to replace DATA statement 0. This statement contains three indicators: the number of data

Program listing.

```
0DATA 5,0, 0
1DATA"ISI - KB/8-79", 0
2DATA"NEWMAN C.E.-  KB/1-79", 0
3DATA"BAMBI INFO - KB/12-78", 0
4DATA"COMP ENTER  - KB/12-78", 0
5DATA"POLYMORPHICS- KB/8-79", 0
999DIML$(1:50),T$(1:25),C$(1:25),X$(1:25)
1000RESTORE0
1010READN,G,Z9
1020IFG=0THEN1100
1050IFG=1THEN1210
1060IFG=2THEN3000
1070IFG=3THEN4000
1100PRINTCHR$(12)\PRINTTAB(15),"LITERATURE REQUESTS"\PRINT
1110PRINT\PRINT\PRINT"CODE"
1120PRINTTAB(3),"1",TAB(10),"CHECK"
1130PRINTTAB(3),"2",TAB(10),"ADD"
1140PRINTTAB(3),"3",TAB(10),"DELETE"
1142PRINTTAB(3),"4",TAB(10),"LIST ALL"
1144PRINTTAB(3),"9",TAB(10),"END OF JOB"
1150PRINT\PRINT\PRINT
1160INPUT"CODE ?",F1
1170IFF1=1THEN1210
1180IFF1=2THEN3000
1190IFF1=3THEN4000
1192IFF1=4THEN5000
1194IFF1=9THEN12000
1195GOTO1160
1210PRINTCHR$(12)
1220PRINT"ENTER REQUEST TO CHECK    HIT RETURN TO EXIT"
1222INPUTT$\IFT$="" THEN1100
1224ON ERROR GOTO1250
1226S1=VAL(T$)\S1=S1+1
1228RESTORE1
1230FORI=1TOS1\READ C$,X\NEXT
1232I=I-1\GOTO1310
1250RESTORE1
1255L3=LEN(T$)
1260FORI=1TON\READ C$,X
1270IF T$=LEFT$(C$,L3)THENEXIT1310
1280NEXT
1290PRINT"NO HIT"
1300GOTO1220
1310I1=I\PRINTCHR$(12)\PRINT"HIT ON ",C$,X
1322PRINT"INPUT 99 TO MARK FOR DELETE"
1330INPUT X\IFX=0ORX=99THEN1350
1340GOTO1310
1350GOSUB1500
1355Z9=1
1360L$=CHR$(48)+"DATA"+STR$(N)+","+CHR$(49)+","+STR$(Z9)
1370Z=FNL(L$)\L$="RUN"\Z=FNL(L$)
1390STOP
1400REM
1430DEFFNL(L$)
1436L$=L$+CHR$(13)
1440FORI=1TOLEN(L$)
1445 A$=MID$(L$,I,I)
1450OUT0,ASC(A$)
1460NEXT
1470RETURN0
1480FNEND
1495REM
1500L$=STR$(I1)+"DATA"+CHR$(34)+C$+CHR$(34)+","+STR$(X)
1510Z=FNL(L$)
1520RETURN
3000REM
3010X=0\I1=N
3020PRINT"ENTER NEW REQUEST/RETURN ... HIT RETURN TO EXIT"
3030PRINT"                  I"
3035INPUTC$
3040IFC$=""THEN1100
3050I1=I1+1\Z9=1
3060GOSUB1500
3070L$=CHR$(48)+"DATA"+STR$(I1)+","+CHR$(50)+","+STR$(Z9)
3080Z=FNL(L$)\L$="RUN"\Z=FNL(L$)
3090GOTO1390
4000REM
4010REM
4020PRINT".....WORKING"
4022RESTORE1
4030D=0
4035FORI=1TON
4038READC$,X
4040IFX=99THEND=D+1
4042NEXT
4045IFD=0THEN4300
4050GOSUB4060
4055GOTO4090
4060PRINT"THERE ARE ",D," DELETIONS"
4070INPUT"SHOULD I CONTINUE (Y OR N)  ?",T$\RETURN
4090IFT$="Y"THEN4100ELSE1100
4100RESTORE1
4110FORI=1TON\READX$,V\NEXT
4120RESTORE1
4130FOR I=1TON\READC$,X
4133IFX=99THENEXIT4150
4140NEXT
4150PRINT"   ",C$,X," BEING DELETED"
4160INPUT"HIT RETURN TO CONTINUE",T$
4170L$=STR$(I)+"DATA"+CHR$(34)+X$+CHR$(34)+","+STR$(V)
4180Z=FNL(L$)\L$=STR$(N)\Z=FNL(L$)
4190Z=N-1\Z9=1
4200L$=CHR$(48)+"DATA"+STR$(Z)+","+CHR$(51)+","+STR$(Z9)
4210Z=FNL(L$)\L$="RUN"\Z=FNL(L$)\GOTO1390
4300L$=CHR$(48)+"DATA"+STR$(N)+","+CHR$(48)+","+STR$(Z9)
4310Z=FNL(L$)\L$="RUN"\Z=FNL(L$)\GOTO1390
5000REM
5010RESTORE2\PRINTCHR$(12)
5020L=0
5030FOR I=2 TO N
5040READC$,X
5045N1=I-1
5050PRINT N1,TAB(5),C$,TAB(35),X
5060L=L+1\IF L>13 THEN GOSUB5500
5070NEXT
5080PRINT\INPUT"RETURN TO CONTINUE",T$\GOTO1100
5500PRINT\INPUT"RETURN TO CONT - X TO ABORT",T$
5510IFT$="X"THEN1100
5520PRINTCHR$(12)\L=0\RETURN
12000REM
12005IFZ9=1THEN13000
12010L$="0DATA"+STR$(N)+","+CHR$(48)+","+STR$(Z9)\Z=FNL(L$)
12020L$="PRINT FREE(0)"\Z=FNL(L$)
12030L$="BYE"\Z=FNL(L$)
12050STOP
13000Z9=0
13010L$="0DATA"+STR$(N)+","+CHR$(48)+","+STR$(Z9)\Z=FNL(L$)
13012L$="EXEC"\Z=FNL(L$)\L$="<2>LIT-R-COM"\Z=FNL(L$)
13040STOP
```

statements used, a code to specify where in the program to branch to and an indicator used at end of job. DATA statement 0 is followed in the buffer with a RUN so that the program will recycle itself back to the add routine. To get from the add routine back to the selection screen, just type a carriage return.

Another function is LIST ALL, which lists all the data in the program. Shown with the data is a record number and a delete indicator. The record number is useful in the CHECK mode.

The CHECK function allows rapid checking of a given company's status. The program requests the name of the company and searches the DATA statements for a match. When a match is found, the program displays all of the data contained within the DATA statement. By using the ON ERROR command in Poly BASIC, I am able to enter just the record number, instead of the full name, of the company, and the display is the same.

To illustrate the ON ERROR command, the program requests the name of the company in statement 1222 INPUT T$. After checking for a null string (T$ = " "), it assumes that I have entered a number and tries to convert it using the function VAL. If it finds an alphabetic character (I entered the name), an error condition occurs because the function VAL is designed to only handle numeric strings. The program remembers that I had coded an ON ERROR statement and then does whatever I specified. In this case, it is a GOTO 1250, where the input string is handled correctly as an alphabetic string. The ON ERROR is a convenient tool not found in all of the extended BASICs.

In the CHECK mode, if a match is found, the program requests that you enter a value for the delete indicator. Normally, you use the CHECK function while reading through a maga-

zine. When an advertisement strikes your eye, you check to determine if a request was made previously of that company for information. If a match is found, just enter a 0, hit return, and the program will continue. If no match is found, then go to the ADD function. When the literature requested arrives, use the CHECK function to get you to the proper record. Set the delete indicator to 99, and when you next use the DELETE function, the record will go away.

The last function is DELETE. Only records with a delete indicator of 99 can be deleted. The program cycles through all DATA statements and counts the number of records to delete. Two checks are built into the program to prevent erroneous deletions. First, after counting, the program requests, "Should I continue?" Any answer other than yes will exit the DELETE function and return to the selection screen. Second, when the specific record is about to be deleted, it is displayed on the screen. A carriage return is necessary to continue the process. This double loop continues until all records with a 99 are eliminated.

Deleted records are not really deleted. If they were, empty spaces would remain in the DATA statements, which would need manual intervention to compress. Rather, a deleted record is replaced by the last DATA statement (its statement number is changed to that of the deleted record), and the last DATA statement number is eliminated. This leaves all the DATA statements compressed at the front of the program.

The last item for discussion is the END OF JOB routine. Because all the information is a part of the BASIC program, you must save the program when you are finished with it or you will lose all the updates. An end of job indicator is built into the program and is part of DATA statement 0. If a record is added, deleted or modified, this indicator sets to 1 from its normal state of 0. At end of job, if the indicator is a 0, the program displays the number of free bytes left and returns to the Poly executive. If the indicator is 1, then the program must be saved.

I utilize another nice Poly feature — invoking a separate command file to compress my diskette, save the updated version of this program, delete the old version, rename the new one and compress the diskette again. Example 2 shows a copy of my command file.

```
; COMMAND FILE FOR LITERATURE-REQUEST
PACK 2
CONTINUE
SAVEF;<2>LIT-R-TEMP.BS
EXEC
DEL <2>LITERATURE-REQUEST.BS
REN <2>LIT-R-TEMP.BS <2>LITERATURE-REQUEST.BS
ZAP
PACK 2
<2>LITERATURE-REQUEST
```

Example 2. Command file.

Conversion to Other BASICs

Most of the program uses standard BASIC commands. The key to conversion to another BASIC is the ability to internally create and output the DATA statements. If you have disks, then the data file can be maintained there.

Conclusion

This program helps to maintain a small data base of information. The possibility exists to use many different codes (besides 0 and 99) in the delete indicator for many different meanings. I have already used this program to handle two different unrelated applications. The program can handle any application that requires maintenance of a list. ■

Questions and Answers On Memory Devices

Part 2 of this series continues with more answers to computerists' questions.

David Price
3901 Victoria Lane
Midlothian, VA 23113

We've come a long way since the days when computer memories were made of relays or vacuum tubes. In this day of rapid inflation, modern memories seem almost anachronistic: every year they become cheaper, faster and more compact. At the same time, memory purchasers are acquiring greater freedom of choice as the market becomes more diverse. But this broader range of products can be a two-edged sword for the newcomer: each new device added to the bushel makes it more difficult to become well informed about the many selections within the bushel.

Some distinctions between product types are pointless and erudite. But there are still many things that the typical newcomer wants to understand. Most of us—when we were tyros ourselves—started by latching onto one of the cognoscenti and asking lots of questions. Others—perhaps inhibited by pride or shame, or both—avoided asking direct questions and took the book or magazine route instead. Maybe you did both. In whatever case, you were hunting for answers to a constant flow of enigmas. We'll start with something simple—just as you did.

Q: Why are there so many different types of memories? Since they all do the same thing, why is there more than one kind?

A: They *don't* all do the same thing. All computer memories have the same basic function—to store information—but beyond that, there are considerable variations.

The most elementary distinction is between *main memory* and *mass storage.* Main memory usually contains the program instructions and data associated with the task that the system is currently handling. Mass storage ordinarily contains programs and data on a long-term basis.

Main memories of contemporary computers (both micro and mini) use *semiconductor* devices. For microcomputers, the most common mass-storage devices are *disk drives* and *tape drives*; in fact, many systems use both. Other types of memory devices are available, too. For now we'll call them *miscellaneous.*

Q: That makes four categories altogether. Should we go through them individually?

A: Agreed.

Magnetic Tape Cassettes

Q: Are the tape cassettes used in microcomputer tape drives the same as those I use for my cassette deck?

A: More or less. In some cases it's best to use special cassettes that are certified for high-density recording. But most hobbyists use standard audio cassettes.

Q: How does the drive itself differ from ordinary tape recorders?

A: Sometimes it doesn't. Often, the drive will actually be a consumer-grade audio recorder adapted for microcomputer use. With most systems, the quality of the drive isn't as important as the reliability of the tape.

Other systems use more advanced drives that can be controlled by electronic commands from the computer. This feature allows the computer direct control of the drive functions.

Q: How can digital information be recorded with audio equipment?

A: An electronic interfacing device between the computer and the tape drive converts the computer output into audio signals. The interface also decodes audio output from the drive and transmits the results to the computer. Thus, the drive can be used both to store and to retrieve information.

Among microcomputers, the most common coding format is the Kansas City Standard, devised in November 1975 by a conference of microcomputer manufacturers. Since there were (and are) many incompatible formats in use, the standard was welcomed because it provided a basis for common interchange of cassettes among users of different equipment.

Q: What are the primary advantages and disadvantages of cassette storage?

A: Compared with disk drives, tape devices sacrifice speed in exchange for economy. Cassette recorders are inexpensive, but their usefulness is limited to *sequential* storage. That is, they can store and retrieve only continuous blocks of data; they cannot access a specific datum except with great difficulty. For instance, if a cassette had 100 numbers stored on it, the tape drive could easily read them all in sequence. It could not, however, zero in on the 53rd number without passing over the first 52. As we'll see in a moment, a disk drive can zero in on one item; this is called *random access* (as opposed to *sequential access*).

Aside from the limitation of sequential storage, the performance of cassette systems is further hindered by their low data-transfer rate. While a slow disk system can read or write data at upwards of 10,000 characters per second, the fastest audio cassette interfaces transfer data at only a few hundred characters per second. Most cassette systems are even slower.

Disks

Q: What is a floppy disk?

A: A floppy disk, or diskette, is the storage medium upon which floppy-disk drives write data and from which they read it. If you could stretch a piece of audio tape like Silly Putty into a disk of a little less than eight inches in diameter, you would have the innards for an 8 inch diskette. Diskettes use the same magnetically sensitive oxide coating as cassettes. The oxide goes on a

heavy Mylar backing, which, after a number of holes have been punched into it, is, in turn, sealed inside a plastic jacket. The jacket helps protect the oxide surface from dust and other contaminants.

Q: How much data can one disk hold?

A: That depends on its size and its format. Eight inch disks that follow the popular IBM 3740 format can hold about 256,256 characters. Smaller disks, 5¼ inches square, commonly hold about 90K.

Many disk systems have provisions for *dual-density* storage, which allows twice as much data to be crammed into the same disk space. Another technique to increase disk capacity is the use of *double-sided* drives and media. A double-sided disk has oxide on both sides of the Mylar; a double-sided drive has a separate read/write mechanism for each side of the disk. This, too, allows twice the normal data capacity. Use of both dual-density formatting and double-sided drives would, naturally, yield quadruple capacity.

In discussions of disk capacity, you must remember whether you are referring to *formatted* or *unformatted* capacity. Since a substantial amount of space throughout the disk is allocated to system-related functions such as identification and control, the capacity available for storage of user data is actually far less than the total disk capacity. The formatted capacity is simply that left over for storage of user data after the overhead disk space is subtracted. The unformatted capacity of a 3740-type disk is around 382,000 bytes, for example, leaving a formatted capacity of 256,256 bytes.

Q: How is data organized on a disk?

A: The disk is first divided into concentric rings called *tracks*. Each track is then sliced like a pie into *sectors*. The disk is not divided physically, of course; these terms merely describe the pattern in which data is stored on the disk surface.

Q: How many tracks does one disk have, and how many sectors does one track have?

A: That, too, depends on the format involved. An 8-inch 3740-format disk is divided into 77 tracks. Each track has 26 sectors. Each sector can hold 128 data bytes. The formatted capacity is, therefore, $77 \times 26 \times 128$, or 256,256. The outermost track is number 0, while the centermost track is number 76.

Q: Number 0? Most people start counting with one, not zero. Why do computer types do it differently?

A: Because . . . well, just because.

Q: Oh, well. What diskette formats are there besides 3740?

A: There are quite a few, actually. One format often used with 5¼ inch minidisks uses 35 tracks, 10 sectors per track, and 256 data bytes per sector—for a total of 89,600 bytes. There exists a more densely packed minidisk format that uses 77 tracks, 16 sectors and 256 bytes per sector—for a total of 315,392 bytes. As you can see, there's a lot of variation.

And there is yet another point of variation to consider: the sectoring method. Some disk formats use *hard sectoring;* some use *soft sectoring.* This distinction involves variation in the construction of the disk itself.

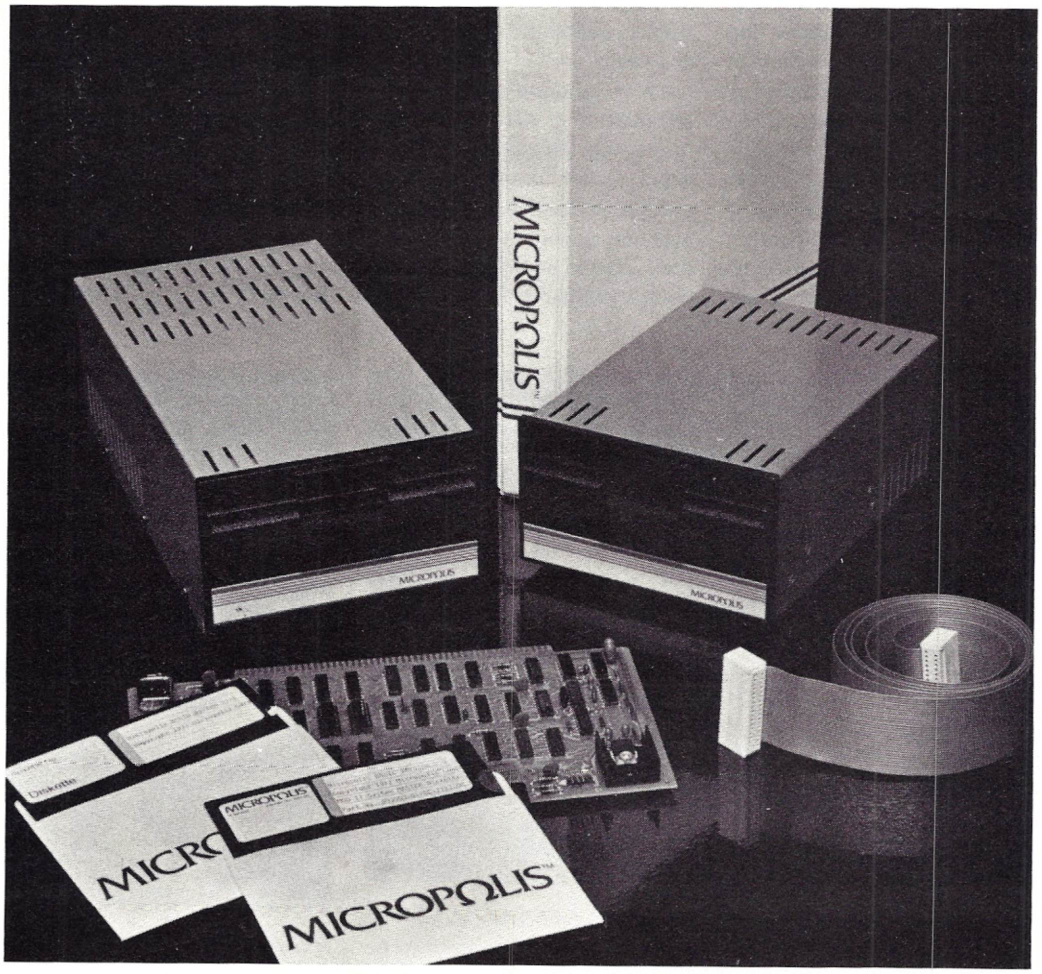
All floppy disks have a hole punched near the outer edge of the Mylar disk. It is called the index hole, whose purpose is to mark revolutions of the disk. Disk drives have sensors that can detect the index hole. When the index hole is detected, the drive controller knows that the sector following the index hole will be sector number zero. The difference between hard sectoring and soft sectoring is this: hard-sectored disks will have, in addition to the one index hole, a sector hole behind every sector; soft-sectored disks have only the index hole. Thus, a hard-sectored disk formatted for ten sectors/track will have eleven holes—one index hole and ten sector holes—punched along its outer edge.

Q: Hard sectoring sounds more complex. What is the advantage?

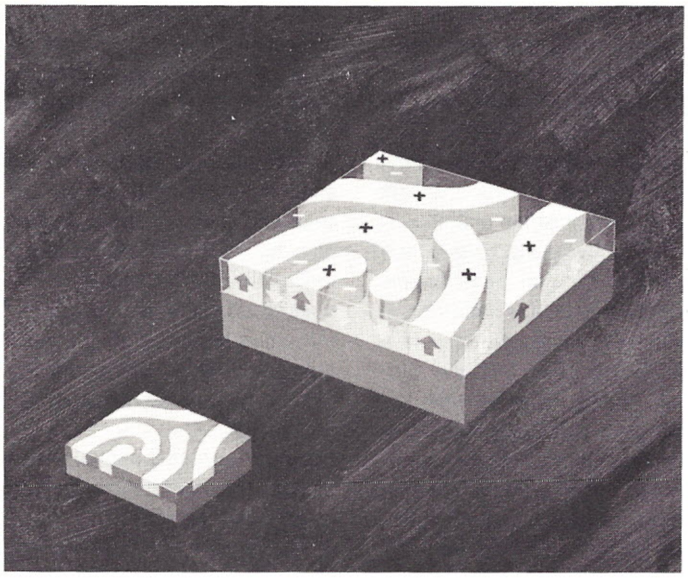
A: Soft-sectored formats must allocate more disk space to overhead, because the absence of a hole to mark each sector means that the sectors must be marked off and identified by information recorded on the disk itself. As a result, soft sectoring reduces the formatted capacity of the disk.

Q: What are the hardware elements that comprise a disk storage subsystem?

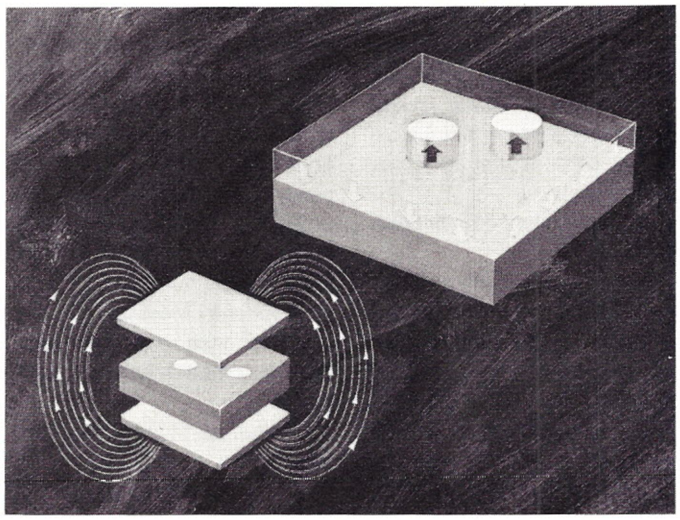
A: The hardware can be divided into two sections: the drive and the controller. The drive includes the mechanical functions, such as the read/write head, a 360 rpm motor to spin the disk within its jacket, a positioner that places the read/write head over the desired track and a photosensor that detects the index hole and the sector holes



A floppy disk subsystem including drives, controller board and diskettes. (Courtesy Micropolis Corp.)



A peek inside a bubble memory. Thin films of certain magnetic materials, such as synthetic garnet, contain randomly shaped domains. *(Photos courtesy of Rockwell International)*



When a magnetic bias field is applied in a direction perpendicular to the thin film by two permanent magnets placed on either side of the device, these randomly shaped domains shrink into bubbles. The bubbles are cylindrical magnetic domains of fixed volume whose polarization is opposite that of the thin film.

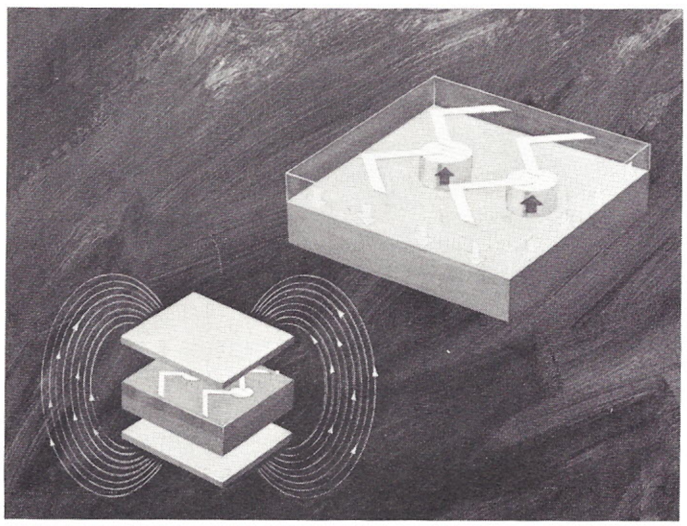
(if any). The controller includes the electronic functions that accept commands from the host computer and translate the commands into instructions that the drive can understand. Some controllers are smarter than others. The smarter the controller, the less work there is for the host computer. Given only the file name, some controllers can retrieve data files, while other controllers have to be told the exact location of the file by track and sector. A few controllers even have their own onboard microprocessors. (Most, however, use high-density integrated circuits designed specifically for the task of controlling a disk drive.)

Q: What about software?

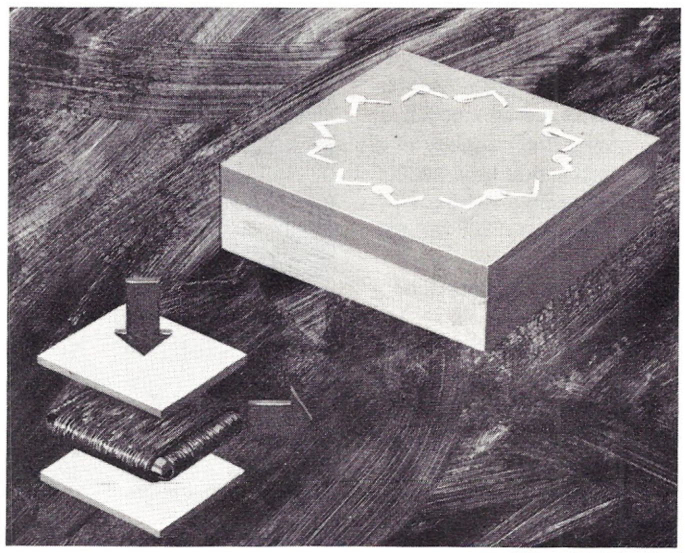
A: The systems software that directly controls drive operations is called the *driver,* which is usually part of a larger program or program series called an *operating system.* (Sometimes the abbreviation *DOS,* which stands for disk operating system, is used instead.) Whenever the operating system has dirty work to be done, it passes control to the driver. Just as the controller translates instructions for the drive unit, the driver is similarly responsible for spoon-feeding the controller, and the operating system translates user commands for the driver.

A bit of overlapping crops up here and there, but the scheme is basically very specialized. And, paradoxically, this specialization allows a good deal of generality in the hardware-software interface. For example, if we had an operating system running on drive "Brand A," and we wanted to use it on "Brand B," we could adapt the entire operating system for the new drive by changing only the driver. The CP/M operating system has become widespread with the help of this approach. The licensee (say, a drive manufacturer) who wants to convert the CP/M operating system to run on his equipment merely modifies a subprogram called *BDOS* (Basic Disk Operating System). BDOS is, among other things, the driver portion of CP/M.

Q: How does an operating system know where to look for a



To guide the movement of the bubbles, a permalloy pattern of chevrons is applied to the surface of the film to form paths. The permalloy paths are formed as loops. The presence of a bubble in a certain position on the loop represents a 1 bit; the absence, a 0 bit.



When, in addition to the magnetic bias field, a rotating magnetic drive field in the plane of the film is applied by a pair of permanent coils, the bubbles move extremely rapidly along the paths.

✓ *Reader Service index—page 241*

given data file?

A: It maintains a directory of files and file locations (by track and sector) on the disk itself.

Q: What if the operating system accidently mangles the directory?

A: Life is like that sometimes. That's why prudent people keep backup copies of important disks.

Q: Are other types of disk storage besides floppy disks available?

A: There are several, but only one has been adopted for microcomputer use: the Winchester design, which differs from floppy drives. First, the disk cannot be removed; it is fixed and sealed within the drive. Second, the disk is typically based on aluminum rather than Mylar; this type of storage therefore has the nickname *hard disk* (as opposed to floppy disk). Third, the read/write head does not touch the disk surface. Unlike the contact-head approach of floppies, Winchesters use a flying head that floats aerodynamically above the disk. The gap between the flying head and the disk is very small, of course, but the absence of direct contact eliminates disk surface wear. Because the disk is protected from both head friction and atmospheric contamination, Winchester drives can also use far greater rotation speeds. The result: faster access and higher capacity. Winchesters are presently available in 8-inch and 14-inch versions with capacities of 10 megabytes (i.e., 10 million characters) and up.

Semiconductors

Q: Why are semiconductor devices used as main memory, while diskettes and cassettes are used as mass storage?

A: Semiconductor memories are faster and more expensive. They are, consequently, well suited for main memories but are too costly for bulk use as mass storage.

Q: How fast are semiconductor memories?

A: Their access times are usually measured in nanoseconds. A nanosecond is a billionth of a second. (A beam of light—no slowpoke by anyone's standards—can travel less than one foot in a nanosecond.)

Speedy semiconductor chips such as the Intel 2125H can cough up a datum in 20 nanoseconds. Most microcomputers, however, use chips in the 200-500 ns range.

Q: What is the difference between programmable memory and read-only memory?

A: Programmable memories can be both read and written. Read-only memories are preprogrammed; the processor cannot directly change their contents.

Q: Read-only memories sound less versatile.

A: They are intended for situations where programmable memories (or RAMs, as they are sometimes called) would be impractical. For example, contemporary RAMs lose their contents when their operating power is interrupted. Thus, whenever a power outage occurs, or whenever the system is turned off, the system must somehow reload all RAM with the desired contents upon power-up. With read-only memory the data is frozen in place and does not need to be reloaded.

Q: How are read-only memories programmed?

A: Different types are programmed in different ways. Mask ROMs, for example, are programmed by the manufacturer. This is economically practical only when you buy a large number (500–1000+) of identical chips at once.

Other types are programmed by the user. The user is then said to be *burning* the device. Since many poeple do not have the equipment needed to burn their own memories, some chip suppliers offer programming services to their customers.

Read-only memories that can be burned only once are called PROMs (programmable read only memory). Some, however, can be erased and reprogrammed. EPROMs (erasable PROM) are erased with ultraviolet light, while EE-PROMs (electrically erasable PROM) can be erased under processor control with electric current.

Q: What is the difference between static and dynamic RAM?

A: Dynamic RAM requires that the system provide periodic electronic signals called *refresh pulses*. Previously, the refresh requirements of dynamic RAM interfered with other functions of the host system. Contemporary designs, however, use *invisible refresh*; that is, they refresh the memories without letting the rest of the system know what's going on. Memory boards designed this way are sometimes called *pseudo static*.

Q: What is the capacity of a single memory chip?

A: Anything you want. You can buy them in nearly any capacity and in any organization. They are specified in terms of pK × q, where p is the depth in units of 1024 bits and q is the width in bits. A 4K × 1 chip, for example, has a total capacity of (4 × 1024) × 1, or 4096 bits. A 2K × 2 chip, although arranged differently, is the same size: (2 × 1024) × 2.

The depth of a chip is the number of addresses it can recognize. The width is the number of bits it outputs upon receipt of a valid address. Thus, a 4K × 1 chip accepts 0 through 4095 as addresses, returning one bit on command. Eight such chips operating in parallel would form a 4K × 8 array. This array would have a capacity of 4K bytes, or 4096 characters. Four such arrays could be integrated to form a 16K × 8 memory board.

Q: How much memory can a microcomputer address?

A: Most are restricted to 64K (65,536 bytes) of directly addressable memory. A technique called *bank switching* is available on some systems as a makeshift way to extend the maximum address space. A system allowing up to eight banks of 64K, for example, could address 512K of main memory.

Q: Can memories make mistakes?

A: All man-made devices have small factory-installed gremlins whose favorite recreation is to make their respective devices fail. We cannot banish the gremlins entirely, but we can use many techniques to mitigate their effects.

One such technique is *parity checking*, which can determine whether a byte is incorrect, but it cannot correct the error. A more complex approach, ECC

A completed bubble memory device photographed on a magnified portion of the chip's circuitry.

(error checking and correction), is based on principles developed by mathematician R. W. Hamming. Unlike parity checking, ECC can consistently detect multibit errors. It can also correct single-bit errors.

Other Memories

Q: What are bubble memories used for?

A: Bubble memories are a compromise in performance between semiconductor and disk memories. They have recently become commercially available.

The bubbles are microscopic magnetic regions that move about inside a thin film. They move synchronously along predetermined tracks so that the chip appears to be a mammoth circular shift register. Current bubble memory chips range in capacity from 92K × 1 (Texas Instruments TIB-100) to 1M × 1 (Intel im7110).

Bubbles have advantages over both semiconductor and disk. Not only are they denser than RAMs, but they are also nonvolatile; that is, they retain their contents even when operating power is removed. In comparison with disk drives, bubbles need less maintenance and protection from contamination, because they have no mechanical moving parts.

Q: Why are core memories no longer in widespread use?

A: They are simply uneconomical. Semiconductor RAM is cheaper and more compact.

Core memories are made of thousands of tiny donuts strung together on crisscrossing wires. The donuts, called *cores*, are made of ferrite or lithium. Interestingly, core memories share several characteristics of bubble memories: they are nonvolatile, slow and require extensive support circuitry.

Q: Are there many other types of microcomputer memory that I ought to know about?

A: Not at the moment, no. But technology is moving fast; tomorrow will doubtlessly bring memories that are cheaper, faster and more compact than ever. Be sure to keep yourself posted. ■

A silicon wafer. The chip on the model's fingertip is similar to those used in semiconductor memory parts.

(Courtesy Rockwell International)

✓ Reader Service index—page 241

Cost-Effective Hard Copy for the Apple II

The system for providing it comprises a Comprint 912 printer, Super-Text and File Cabinet.

Allan Turoff
229 E. 53rd St.
New York, NY 10022

A few months ago, while sitting in front of my Apple II, I decided to add hard copy to the system. I settled on a Comprint 912 by Computer Printers International; Super-Text by the Muse Co.; and File Cabinet, available from the Apple Software Bank of Contributed Programs. This combination of printer and programs gives me a low-cost yet effective hard-copy system that meets my needs better than any other currently available package.

The Comprint 912 is among the least expensive printers, listing at $660, but probably available for less. It is fast—225 characters per second—yet its print quality is surprisingly good because the letters are formed from a 9×12 dot matrix, which gives superior definition to letters. (See the review of this printer in the March 1980 issue of *Microcomputing.*)

At about $100, Muse's Super-Text was double the price of their Dr. Memory word processor, but because it furnished so many more controls, it was easily worth the difference. The single and double keystrokes are easy to remember because the letters usually stand for what commands do (e.g., "b" means go to bottom screen, "t" means go to top screen).

Super-Text has standard screen features: insert and delete characters; shift incomplete words at the end of a line to the next line; move cursor all over screen; scroll up or down, a line or page at a time, to beginning or end of text, or to the position of the last change made. But Super-Text also has three unique features.

You can split the screen into two parts, and even vary the screen sizes. This allows you to keep notes or instructions on one screen while editing on the other. You can compare the old sentence with the newly edited sentence or you can look something up in one part of the file while working in another.

Super-Text has a built-in math module that furnishes automatic totaling of columns and computation of formulae. You can select the number of decimal places, from 0 to 9. Super-Text switches to scientific notation when necessary and has 14 significant digits.

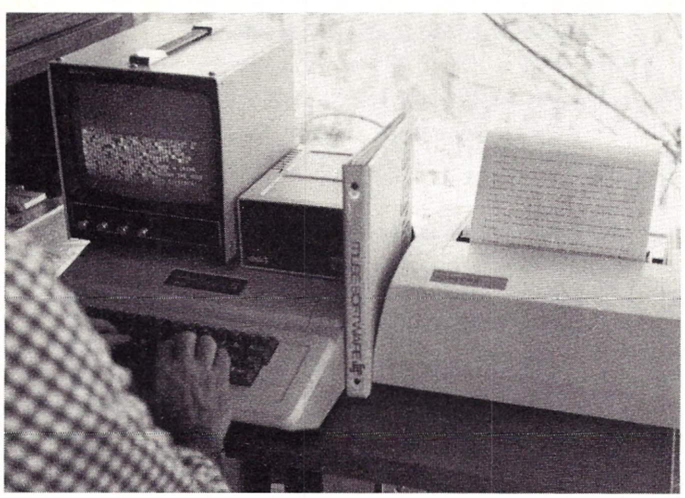
A feature called Autolink allows you to connect an unlimited number of files when printing or when searching or searching and replacing. For instance, say you had two files of names, addresses and phone numbers called "LIST1" and "LIST2," and you wanted to find a phone number that might be in either file. Through Autolink you'd load the first file and do a "find." Super-Text would search both files, if necessary, until the desired information was found. This also works with multiple disk drives, so you can imagine Super-Text searching for information through dozens of files on each of several disks.

There are other features you'll appreciate as you work with Super-Text, such as the option of printing the word "the" with one keystroke. This is a time-saver because the word appears so frequently. The search-and-replace feature is also a time-saver because it allows you to type a short abbreviation for a long phrase that occurs often in your text, then replace the abbreviation with the full phrase.

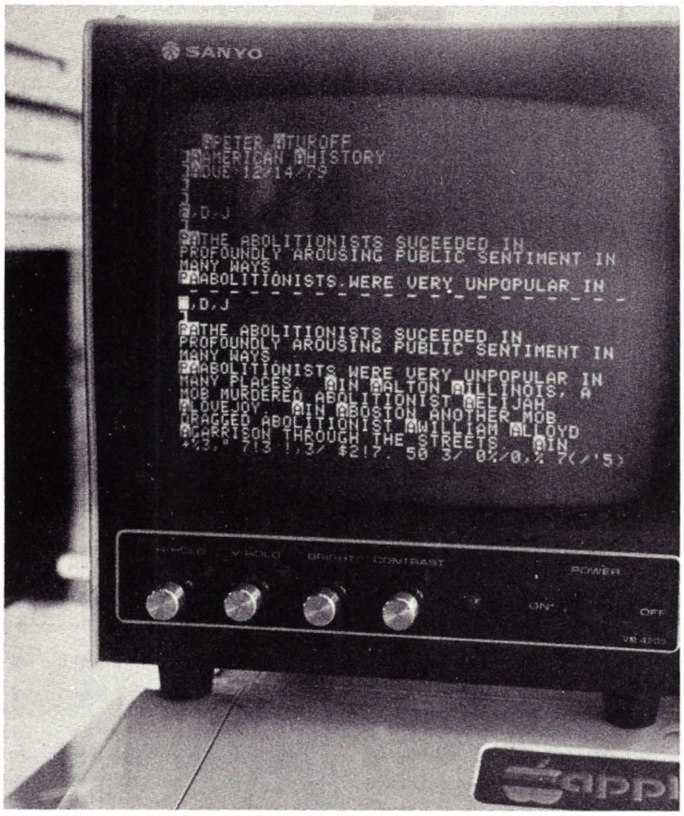
Another nice feature is loading files by number so you don't have to type in a long file title, which may be mistaken or misspelled, resulting in a file-not-found message. Other features are tabbing and right- or left-justification of columnar numbers or text, block moves, multiple copies, numbering and centering.

Combining the Comprint 912 with the Muse's Super-Text resulted in a cost-effective under-$800 limited, "semi-automatic" typewriting system. I'm balancing low cost, quick listing ability and fairly good print quality vs some limitations in printing capability and quality. The quality is good, but it cannot equal that of Diablos or Qumes. However, these machines cost five times as much as the Comprint 912 and type at one-fourth the speed. It's not easy to do envelopes, forms or a large mailing, but the 912 is capable of producing letters, reports, memos, birthday cards, birth announcements, game rules, recipes and personal finance reports.

The structured Applesoft data-base management system



The system, set up and working.



Note Super-Text split screen on monitor.

called File Cabinet, available from the Apple Software Bank, allows you to set up your record structure with as many or as few fields as you wish. You can sort, add or delete records. Also, you can set up a format to print portions of the data in a variety of forms. It can total columns of figures, both horizontally and vertically. With the aid of the printer, I listed the program and found that it was not too hard to modify to incorporate improvements or custom commands.

The active user group A.P.P.L.E. has an improved version with a faster sort routine, among other improvements. I use it to analyze my financial expenditures and as a name/address/phone/birthday/anniversary file; I plan to use it for a disk-program master catalog. In my opinion, File Cabinet is one of the finer data-base-management programs available at present.

Overcoming Problems

Although my Comprint 912 tested OK, it would not print anything that came from the computer. After a call to the factory, I decided to ship everything back. Within ten days the printer went coast to coast, was fixed and returned. The trouble was in the wiring of the parallel interface card, the fault of my local computer store (by the way, Comprint 912s now come with a triple interface for Apple-, TRS-80- or Centronics-compatible connections.

At the factory the Comprint people noticed that the paper occasionally wrinkled and discovered a faulty paper-advancing roller. They replaced it, rewired my interface and returned it all.

My joy was curtailed when I ran through the 100 feet of paper supplied with the printer in about 3.14 days, and further curtailed when I learned that the local computer stores did not yet carry paper. I finally obtained paper through the recommended paper factory.

In spite of minimal problems, these three components give me hard-copy output of program listings, letters, articles, reports and records. I am pleased with the system. ■

Ross A. Wirth
15906 E. 96 St. N.
Owasso, OK 74055

Matrix Sorting

Don't get stuck with "single sorting." Use this technique for multiple data sorts.

A few articles have been written lately on sorting. As any programmer knows, sooner or later you will need to sort some data. But, alas, all the sort programs presented so far deal with sorting only one array.

What if you have more than one type of data—all related—that must be sorted? What do you do? Sort the one array and do the same swaps on all the other arrays as you do on the first? No. You use a matrix sorting subroutine!

The matrix sorting subroutine in Level II TRS-80 BASIC is generalized for sorting two-dimensional arrays by columns or rows. This subroutine employs two versatile techniques used by many programmers, especially technical programmers: matrix storage of data and secondary indexes.

A Case in Point

Before getting into the subroutine itself, let's look at two examples that will illustrate the benefits available from matrix sorting. The first example is an actual case, though greatly simplified for ease of presentation, of a regional sales manager who was responsible for the operation of four stores in his region. To help him in his job he wrote a program to handle the calculation of the profit and loss statement for each store.

He had his data stored in a number of arrays by store-number sequence. There was an array for sales and an another array for expenses. (Actually, there were over 50 separate items collected this way for twelve stores.) All references to store number were by hard coding, that is, if the first item in the array was being processed and printed, the program would print the store number by executing the following statement: IF I=1 THEN PRINT "1254";.

Naturally, coding such as this greatly increased the length of the program. You really cannot fault this program design, though, because it did work. However, the regions were just reorganized, and two of the stores in his region were now in a new region. He also now had responsibility for two new stores. Not only did he have to become acquainted with two new stores, but he had to change his program references from the old store numbers to the new store numbers.

Required changes such as this are not at all uncommon when you do excessive hard coding rather than designing your program to be table driven. This store manager also had another problem with his existing profit and loss program: he was limited to one form of presentation and had to do extensive reprogramming to alter the report in any way.

The second example concerns a local retired businessman and his attempts to keep track of his stock portfolio. His program was much like the regional sales manager's. There was a lot of hard coding of data and little flexibility in reporting. He had the ability to print the report in alphabetical sequence (the way his data was stored) or in current dividend yield sequence.

The sort subroutine for sorting by dividend yield had the data swapped in all arrays, along with the array of dividend yields. Once sorted, though, the data was no longer in alphabetical sequence.

	Secondary Index	Array to be sorted
First pass	1 2 3 4	21 13 11 18 swap
	2 1 3 4	21 13 11 18 swap
	2 3 1 4	21 13 11 18 swap
Second pass	2 3 4 1	21 13 11 18 swap
	3 2 4 1	21 13 11 18
Third pass	3 2 4 1	21 13 11 18
final	3 2 4 1	21 13 11 18

Table 1. Use of secondary indexes in sorting.

Variable	Subroutine Information Flow	Description
N	input	number of items to be sorted
A(,)	input	two dimensional array, one dimension of which is to be sorted
S( )	output	subscript array (secondary indexes)
RC	input	control switch 1 row sort 2 column sort
I	input	row number to determine sort order in column sort
J	input	column number to determine sort order in row sort
II	local	subscript
C	local	counter
N1	local	remaining items to be sorted
X	local	swap variable

Table 2. Variable usage in matrix sort.

The solution in both cases involved organizing the data into a matrix and using the generalized row-column sort subroutine shown in Program A. A simplified version of the program developed for the regional sales manager is shown as Program B. Program C is the program developed for the retired businessman. Both applications use the same subroutine—one for sorting the rows of a matrix, the other for sorting the columns in the matrix.

```
10 REM EXAMPLE OF GENERAL COLUMN SORT
20 CLS
30 DIM A(4,10),S(4)
40 REM  STORE #, SALES, EXPENSES
45 N=4
50 DATA 1254,    8792,  6510
60 DATA 2871,    7641,  6249
70 DATA 5942,    8532,  7153
80 DATA 4280,    6998,  4920
90 FOR J=1 TO N
100  FOR I=1 TO 3
110    READ A(I,J)
120  NEXT I
130 NEXT J
140 REM CALCULATE INCOME
150 FOR J=1 TO N
160   A(4,J)=A(2,J)-A(3,J)
170 NEXT J
180 PRINT "UNSORTED REPORT"
190 PRINT
200 FOR I=1 TO N
210   S(I)=I
220 NEXT I
230 GOSUB 1000
240 PRINT:PRINT:PRINT
250 PRINT "SORTED ON STORE NUMBER"
260 PRINT
270 RC=2
280 I=1
290 N=4
300 GOSUB 4000
310 GOSUB 1000
320 PRINT:PRINT:PRINT
330 PRINT "SORTED ON SALES"
340 PRINT
350 REM RC & N STILL SET FROM ABOVE
360 I=2
370 GOSUB 4000
380 GOSUB 1000
390 PRINT:PRINT:PRINT
400 PRINT "SORTED ON EXPENSES"
410 PRINT
420 REM RC & N STILL SET FROM ABOVE
430 I=3
440 GOSUB 4000
450 GOSUB 1000
460 PRINT:PRINT:PRINT
470 PRINT "SORTED ON INCOME"
480 PRINT
490 REM RC & N STILL SET FROM ABOVE
500 I=4
510 GOSUB 4000
520 GOSUB 1000
530 END
1000 REM PRINT REPORT
1010 PRINT "STORE NUMBER";
1020 I=1
1030 GOSUB 2000
1040 PRINT
1050 PRINT "SALES";
1060 I=2
1070 GOSUB 2000
1080 PRINT "EXPENSES";
1090 I=3
1100 GOSUB 2000
1110 PRINT
1120 PRINT "INCOME";
1130 I=4
1140 GOSUB 2000
1150 RETURN

2000 REM PRINT REPORT LINE FROM A(I,S(J))
2010 FOR J=1 TO N
2020   PRINT TAB(11*J+2) A(I,S(J));
2030 NEXT J
2040 PRINT
2050 RETURN
4000 REM GENERALIZED ROW-COLUMN MATRIX SORT
4010 FOR II=1 TO N
4020   S(II)=II
4030 NEXT II
4040 N1=N
4050 C=0:N1=N1-1
4060 IF N1=0 THEN 4160
4070 FOR II=1 TO N1
4080   IF(RC=1)AND(A(S(II),J)<=A(S(II+1),J)) THEN 4140
4090   IF(RC=2)AND(A(I,S(II))<=A(I,S(II+1))) THEN 4140
4100   X=S(II)
4110   S(II)=S(II+1)
4120   S(II+1)=X
4130   C=1
4140 NEXT II
4150 IF C>0 THEN 4050
4160 RETURN
```

Program B. Column sort.

How It Works

The core of the matrix sort is a modified ripple sort. If very large arrays are being used, the sort technique can easily be changed to one of the tree sorts. There are really only two differences between this sort and a regular modified ripple sort.

First, there are two IF tests, rather than one. The first IF (line 4080) is used when rows are being sorted; the second IF (line 4090) is for column sorting. Only one is used during the sort, but by including both, the sort gains another dimension (literally).

The second difference from a regular modified ripple sort involves the swap and references to the subscripts. The new code is for the secondary index. Rather than using II as the subscript of array A, the II^th^ value of S is used. Thus, the swap does not move any data in matrix A but does move the subscripts stored in S. The stock portfolio program (Program C) has an additional test included for alphabetizing the stocks (line 4095). The subscripts in S can then be used for ordering matrix A as well as the array of stock names.

An additional benefit in using secondary indexes is the reduction in the amount of data to be swapped; the original data is not moved. Table 1 shows a simple example of how a secondary index is used. Table 2 lists the variables referenced in the matrix sort subroutine and how they are used.

Program B also shows an additional benefit of using matrixes to hold program data: generalized output subroutines. To print any report, the same subroutine is used (lines 1000–1150). To print any row in any report, another subroutine

```
4000 REM GENERALIZED ROW-COLUMN MATRIX SORT
4010 FOR II=1 TO N
4020   S(II)=II
4030 NEXT II
4040 N1=N
4050 C=0:N1=N1-1
4060 IF N1=0 THEN 4160
4070 FOR II=1 TO N1
4080   IF(RC=1)AND(A(S(II),J)<=A(S(II+1),J)) THEN 4140
4090   IF(RC=2)AND(A(I,S(II))<=A(I,S(II+1))) THEN 4140
4100   X=S(II)
4110   S(II)=S(II+1)
4120   S(II+1)=X
4130   C=1
4140 NEXT II
4150 IF C>0 THEN 4050
4160 RETURN
```

Program A. General row-column matrix sort.

Program C. Row sort.

```
10 REM EXAMPLE OF GENERAL ROW SORT
20 CLS:DIM A(100,8),S(100),DC(8),DF(4),SN$(100)
30 N=10
40 REM STOCK NAME,      PRICE,  DIV,  E/S,  SHARES
50 DATA CITCO,          62.75,  3.2, 12.07, 500
60 DATA XEROX,          57.875, 2.4,  1.66, 0
70 DATA DEERE,          35.375, 1.6,  2.74, 1000
80 DATA WARNER-LAMBERT, 24,     1.2,   .72, 20
90 DATA K MART,         25.875,  .84,  .37, 280
100 DATA TEXACO,        26.75,  2,    1.13, 700
110 DATA MURPHY,        14.625, 1,     .6,  100
120 DATA TORO,          21.5,    .72, 2.43, 0
130 DATA MONSANTO,      49,     3.2,  4.4,  100
140 DATA FIELDCREST,    28.625, 1.4,  1.11,  30
200 FOR I=1 TO N
210   READ SN$(I)
220   FOR J=1 TO 4
230     READ A(I,J)
240   NEXT J
250 NEXT I
260 DATA ########,######.#,#####.##,####.###
270 FOR I=0 TO 3
280   READ DF$(I)
290 NEXT I
300 DATA 3,2,2,0,1,1,0,2
310 FOR I=1 TO 8
320   READ DC(I)
330 NEXT I
340 REM CALCULATIONS
350 FOR I=1 TO N
```

```
360   REM CALCULATE DIVIDEND RATE OF RETURN  A(I,5)
370   A(I,5)=A(I,2)/A(I,1) * 100
380   REM CALCULATE PRICE/EARNINGS RATIO     A(I,6)
390   A(I,6)=A(I,1)/A(I,3)
400   REM CALCULATE MARKET VALUE             A(I,7)
410   A(I,7)=A(I,1)*A(I,4)
420   REM CALCULATE ANNUAL DIVIDEND RECEIVED A(I,8)
430   A(I,8)=A(I,2)*A(I,4)
440 NEXT I
500 CLS
510 PRINT "ENTER REPORT SORT OPTION"
520 PRINT TAB(10) " 1.  NO SORTING"
530 PRINT TAB(10) " 2.  ALPHABETIZED"
540 PRINT TAB(10) " 3.  PRICE RANKING"
550 PRINT TAB(10) " 4.  DIVIDEND RANKING"
560 PRINT TAB(10) " 5.  EARNING/SHARE RANKING"
570 PRINT TAB(10) " 6.  # OF SHARES OWNED"
580 PRINT TAB(10) " 7.  DIVIDEND RATE OF RETURN"
590 PRINT TAB(10) " 8.  PRICE/EARNINGS RATIO"
600 PRINT TAB(10) " 9.  MARKET VALUE OF SHARES OWNED"
610 PRINT TAB(10) "10.  ANNUAL DIVIDEND RECEIVED"
620 PRINT TAB(10) "11.  END"
630 PRINT
640 FOR I=1 TO N
650   S(I)=I
660 NEXT I
670 INPUT"ENTER OPTION";O
680 IF O<>INT(O) THEN 500
690 IF(O<1)OR(O>11) THEN 500
700 IF O=11 THEN END
710 IF O=1  THEN 750
720 IF O=2  THEN RC=3 ELSE RC=1
730 J=O-2
740 GOSUB 4000
750 PRINT
760 INPUT"TURN ON LINE PRINTER AND HIT ENTER";Z$
770 IF O<3 THEN 780
771 LPRINT TAB(18) " ";
772 FOR I=1 TO O-2
773   LPRINT "         ";
774 NEXT I
775 LPRINT "*"
780 LPRINT"  STOCK";
790 LPRINT TAB(20) "   PRICE";
800 LPRINT"  DIV/SHR";
810 LPRINT"   E/S ";
820 LPRINT"  SHARES";
830 LPRINT"   DIV%RET";
840 LPRINT"   P/E";
850 LPRINT"   VALUE";
860 LPRINT" TOT DIV"
870 LPRINT "         "
880 FOR I=1 TO N
890   LPRINT SN$(S(I));
900   LPRINT TAB(19) " ";
910   FOR J=1 TO 8
920     LPRINT USING DF$(DC(J)); A(S(I),J);
930   NEXT J
940   LPRINT " "
950 NEXT I
960 LPRINT
970 LPRINT
980 LPRINT
990 GOTO 500
4000 REM GENERALIZED ROW-COLUMN MATRIX SORT
4010 FOR II=1 TO N
4020   S(II)=II
4030 NEXT II
4040 N1=N
4050 C=0:N1=N1-1
4060 IF N1=0 THEN 4160
4070 FOR II=1 TO N1
4080   IF(RC=1)AND(A(S(II),J)<=A(S(II+1),J)) THEN 4140
4090   REM IF(RC=2)AND(A(I,S(II))<=A(I,S(II+1))) THEN 4140
4095   IF(RC=3)AND(SN$(S(II))<=SN$(S(II+1))) THEN 4140
4100   X=S(II)
4110   S(II)=S(II+1)
4120   S(II+1)=X
4130   C=1
4140 NEXT II
4150 IF C>0 THEN 4050
4160 RETURN
```

is used (lines 2000–2050). If any report formatting needs to be redone, all the code will be in one place, and the changes can easily be accomplished.

Conclusion

The matrix sort subroutine presented here is easy to use and versatile. When writing your next program, incorporate the matrix sort and use a matrix to hold your data. And don't forget about secondary indexes. Once you get used to using these techniques, you will never go back to your old programming techniques. I never will! ■

✓ Reader Service index—page 241

Disk Divinations

How to build and use a small tester for mini-floppy disk drives.

William Hosking
8626 E. Clarendon
Scottsdale, AZ 85251

Since I repair mini-floppy disk drives in my spare time at home, it didn't take me long to realize it was tremendously inconvenient to connect and disconnect from my computer system every time I wanted to check a different drive's functions. I set out to design a small test fixture to test all of the major functions on a standard mini-floppy drive without the use of a computer. Since the disk drive connections are standardized, the logic and power connect easily and only require one type of plug each.

Design Concept

The primary functional signals to and from a mini-floppy are listed in Table 1. These pinouts have been standardized, at least for Shugart and Wangco (now Siemens). A complete schematic of my test box is shown in Fig. 1.

The incoming signals to the test box from the drive are: write protect (WP), track 0 (TR00), index and read data. They are open collector outputs from the drive and require 150 Ohm terminations to +5 V at the test box. The signals are then fed into a 4049 inverting CMOS buffer, which, in turn, drives LEDs on the panel of the test box. Although both index and read data are pulsed signals, when operating properly, they will provide an adequate indication on the LED.

The outputs from the test box to the drive are almost all switch closures. The drive selects and motor control are active when grounded, so a simple switch closure to ground will do the job. Switch debouncing is not required.

The direction line is on a switch, where the on (ground) position represents step in and the open position is step out. The write gate output is grounded for write enable and open for read enable.

The remaining two outputs from the test fixture are pulses. The step button produces a negative going pulse of approximately 2 microseconds to step the head in whichever direction has been selected by the direction switch.

I have hooked up the second side of the multivibrator as free running to put a series of pulses on the write data input. This will not write anything usable to a disk, but it will provide a digital pulse stream for tracing through the disk write circuitry. As a final touch, I put a BNC jack on the test box to bring out the read data signal for observation on an oscilloscope.

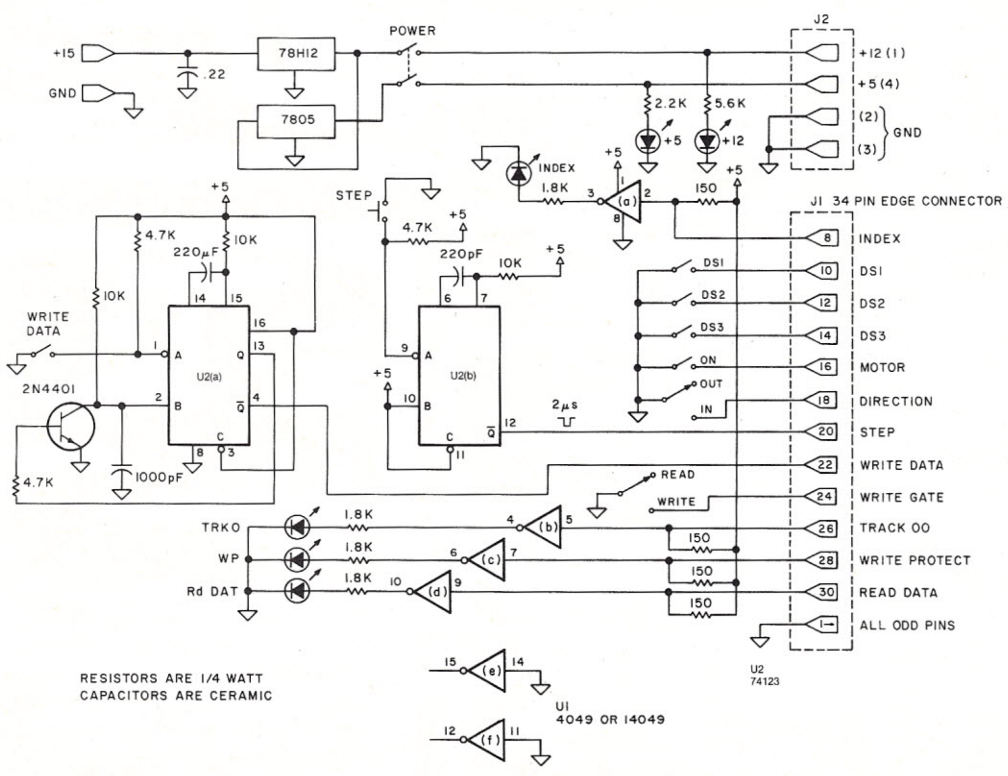
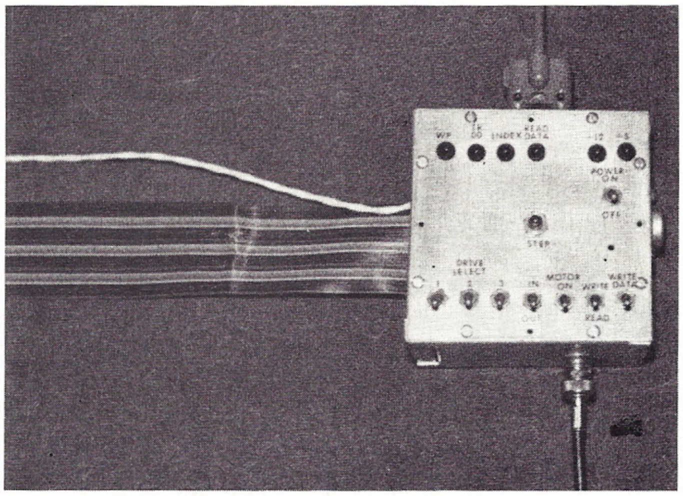


Fig. 1. Schematic diagram of the mini-floppy drive test fixture.

Operation

Utilization of the drive test box makes operational checkout of a mini-floppy disk drive simple. Simply connect power and signal connectors via the appropriate cables, supply power to the test fixture and continue.

Detailed troubleshooting of a drive is beyond the scope of this article; however, I have presented the following steps as a basic guide to using the test fixture. One of the three drive



Mini-floppy drive test box.

Pin	Signal	Notes
1-33 odd	ground	
2,4,6	no connection	
8	Index	Output, open collector
10	DS1	Input, ground active
12	DS2	Input, ground active
14	DS3	Input, ground active
16	Motor	Input, ground for on
18	Direction	Input, ground = in, open = out
20	Step	Input, pulse negative
22	Write Data	Input, negative pulses
24	Write Gate	Input, ground = write
26	Track 00	Output, open collector
28	Write Protect	Output, open collector
30	Read Data	Output, open collector
32,34	no connection	

Table 1. 5¼ inch mini-floppy disk drive edge connector signals.

select lines should cause the drive to activate as indicated by the front panel LED on the disk drive. Once the drive is selected, turning on the motor switch should cause the motor to start.

If a disk had been loaded, the index LED should start flickering, and the read data LED should be on. Additionally, if the disk inserted was write protected (notch taped over), the WP LED should now be on. If an oscilloscope is connected to the BNC connector on the test fixture, it should show the read data output pulses.

Now observe the sliding read-write head assembly through the side of the drive and momentarily push the step button on the test fixture. The slide should step one position in whatever direction the direction switch is set. Track 0 is the outermost track position. Setting the switch to out and repetitively pushing the step button should cause the head to move out until the track 00 LED, which is the reference point for all disk operations, comes on.

Before putting a non-protected disk in and turning on the write data and write gate, make sure that you don't damage any data, since it will cause erasure.

Conclusion

This device is convenient for repairing mini-floppies. To finish the package, however, you need schematics for the various models and versions of drives you intend to service. These are usually part of a service manual that can be obtained from a distributor or company rep. ■

Colorful Baud-Rate Generator

An inexpensive TV color-burst crystal provides the basic frequency for a counter chain producing UART clock frequencies for ten standard baud rates.

Ken Barbier
Borrego Engineering
PO Box 1253
Borrego Springs, CA 92004

UART" and "USART" and other combinations of the same letters refer to integrated circuits that are universal, synchronous, and/or asynchronous, receiver/transmitters. Their internal makeup is described by the particular combination of these letters. One manufacturer refers to his device simply as an "SIO" for serial input/output.

No matter how designated, the devices are used for the serial transmission of digital data. The transmitter section takes a byte of data in parallel, appends start and stop bits and a parity bit, if desired, and shifts out the data one bit at a time at one of many standard signaling rates, referred to as the "baud" rate.

The receiver section of the same device is used to gather a stream of serial data bits, strip off the start and stop bits, check the parity of the received data and present it to the receiving device in a parallel format.

With many of the UARTs now on the market you can transmit (Tx) and receive (Rx) at the same time; with some of these ICs, the Tx and Rx can be at different baud rates at the same time. The Tx and Rx clock inputs to the UART are at 16 times the baud rate (16 ×), and it is the clock frequency that determines at what rate the transmitter will shift out the data bits. If the Rx clock is not the proper 16 × frequency, the received data is garbled.

The clock frequency supplied to the UART must be within 5 percent of the correct value for the desired baud rate. The baud-rate generators on the market use a crystal oscillator and digital counters to produce the desired 16 × clock. Their accuracy is well within 1 percent, which is many times the accuracy required.

One disadvantage of the IC baud-rate generators is that they can produce only a single frequency at a time and must be switched from baud rate to baud rate when different signaling rates are desired. In a microcomputer system, it may be desirable to have several UARTs running simultaneously at different rates.

For instance, in my system I send data to a line printer at 1800 baud, send and receive from an audio tape interface at 600 baud and operate an ASR-33 Teletype at 110 baud. To avoid having to switch baud rates or patch interfaces, each device has its own UART, which simplifies things and reduces operator errors.

A single generator (Fig. 1) produces all the standard 16 × clock

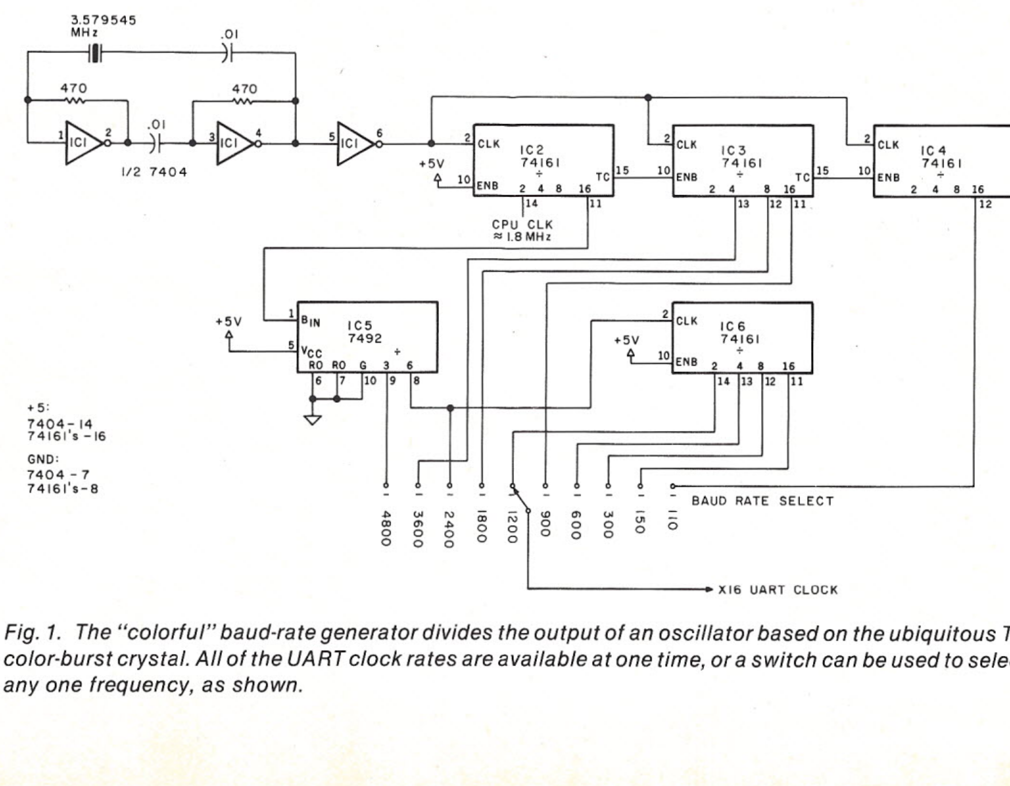


Fig. 1. The "colorful" baud-rate generator divides the output of an oscillator based on the ubiquitous TV color-burst crystal. All of the UART clock rates are available at one time, or a switch can be used to select any one frequency, as shown.

frequencies corresponding to baud rates from 4800 to 110. Although the schematic shows a selector switch to pick off the desired rate, I don't use one in my system since all the clocks are produced simultaneously, and my UARTs are each dedicated to a particular baud rate.

Producing *all* of these rates in a particular system is probably not necessary, so the circuitry shown can be simplified. For example, if you could operate your remote terminal at 3600 baud, and needed only one other rate for a TTY printer (110 baud), you could eliminate IC5 and IC6. Omitting IC3 and IC4 would still produce the binary sequence of rates: 4800, 2400, . . . 150.

The ICs and the crystal chosen for this circuit are all common types available at "jelly bean" prices. The crystal is the TV color-burst frequency, 3.579545 MHz, and is available at prices ranging down to 99 cents. This means the circuit can be built for less than the cost of a dedicated baud-rate generator IC and its special frequency crystal. When completed, this circuit produces all the frequencies at the same time.

You could even tap off 3.579545 or 1.789773 MHz for use as a microprocessor CPU clock, but then those guys running at 4.00 or 2.00 MHz would beat you out in the *Kilobaud* benchmark race (see the June and October 1977 issues).

Accuracy Considerations

Table 1 lists the baud rates, the desired 16 × clock rates and the actual frequencies produced by this circuit, rounded off to the nearest 1 Hz. The Percent Error column shows that, depending on the path taken through the counters, the clock rates are in error either 0.7 percent or 2.9 percent, all on the low side.

How accurate does your UART clock have to be? Since you can assume that the equipment you are communicating with uses the more accurate baud-rate generators, you can hog almost the entire error budget on your end. Fig. 2 illustrates the allowable error.

The UART detects the leading edge of the start bit, then spaces over one-half of one bit duration and samples the start bit to make sure it is still there. Because of this technique, a transient spike would have to be at least one-half bit long to fool the UART. Since we have supplied the UART with a clock at 16 times the bit rate, the most accurately it can position its sample point is ± 1/16 of the bit time, or 6.25 percent. The UART is aiming for the center of the bit time, but can miss the 50 percent point by as much as 6.25 percent. This leaves us an error budget of 43.75 percent of one bit time. All successive samples of the following bits must be taken within this window.

Baud Rate	UART Clock	Actual Clock	Percent Error
4800	76800	74574	2.9
3600	57600	55930	0.7
2400	38400	37287	2.9
1800	28800	27965	0.7
1200	19200	18643	2.9
900	14400	13983	0.7
600	9600	9322	2.9
300	4800	4661	2.9
150	2400	2330	2.9
110	1760	1748	0.7

Table 1.

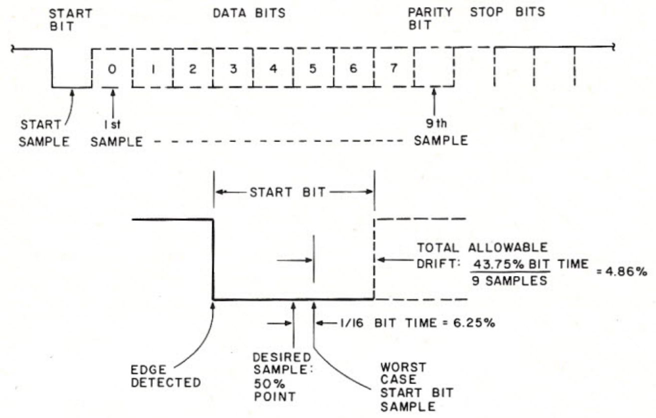


Fig. 2. The "worst case" condition of eight data bits and a parity bit. If the received UART clock rate is in error, the sample point will drift through the bit stream. To insure proper sampling of the last (parity) bit, the clock must be within 4.85 percent of the proper frequency.

As you can see from Fig. 2a, you have nine more samples to take. Since your colorful clock generator is stable, but not accurate, you have to be sure that your frequency is within 43.75 percent divided by nine samples, or 4.85 percent of the desired value. At a maximum error rate of 2.9 percent, you are safe.

You're not too safe, though, as other factors contribute to errors. Any asymmetry in the circuits that decode and process the serial bit stream before it gets to the UART can cause trouble. In actual practice, the colorful baud-rate generator has proved capable of providing error-free communications through 20 mA current loops, RS-232 paths and the modem used for recording data on audio cassette recorders.

This one generator can run them all at the same time! ■

Beat the MIKBUG Memory Squeeze

Select between monitor and memory at the flip of a switch.

Bill Vodall
Box 336
Oilmont, MT 59466

I appreciate the convenience of having a power-on monitor such as MIKBUG, but don't believe it is worth the 8K of the memory map required in addition to preventing use of the reset and software interrupt vectors of the 6800. By adding 3 ICs I was able to have MIKBUG and memory too. Now I have a switch on the front panel that will give either MIKBUG from \$E000 through \$FFFF as normal, or in the other position will move a 4K static memory board from \$7000 to \$F000. This gives me memory in the uppermost position of the memory map as well as access to all the 6800 vector addresses.

The Procedure

I used the two user-defined lines on the SWTP bus, but jumper wires could be used. Remember that this involves rerouting an address line and chip enables, so keep everything short and neat.

Step 1: Disconnect pin 8 of IC16 on the SWTP MP-A processor board by either pulling it out of the socket or unsoldering it from the circuit board. Then run a jumper from pin 8 on the board directly down to the pad for UD1. See Fig. 1.

Step 2: Cut the trace from A15 on the Molex connector to pin 4 of IC22 on an MP-M 4K memory board. Run a jumper from the IC22 side of the cut to the UD2 pad. The memory board can only be enabled when UD2 is brought low. See Fig. 2.

Step 3: Build the circuit of Fig. 3. I used a Seal WireWrap board, but use whatever technique you prefer. Remember—keep everything short. Now whenever the select line is high (a plus 5) everything will appear normal with greedy MIKBUG taking its 8K of memory, and the 4K RAM board will be in the lower half of memory at its normally addressed position.

Pull the select line to ground and the RAM board will jump 32K higher; half of MIKBUG will disappear. Please note that the RAM board must be addressed to \$7000 for it to take the place of the upper 4K of memory.

Step 4: Decide what type of circuit to use to control the select line. See Fig. 4. I used circuit A because I didn't need any fancy control. Circuit B or C can be used with a memory-decoded strobe signal (pin 1 of unused I/O ports) or control characters from a terminal such as a CT-64. Any negative-going pulse 2 or more microseconds long will work. The switch in circuit C can be used to prevent damage from wild programs during the debugging stage.

Now I can turn on the computer, use MIKBUG to load a program into the RAM board at \$7000, flick the switch, and I have full control. One nice thing about this memory modification is that all of the MIKBUG rou-

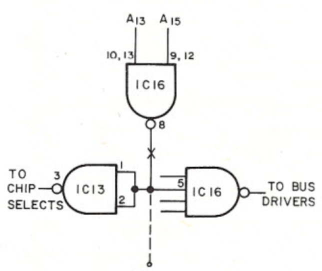


Fig. 1. CPU mod.

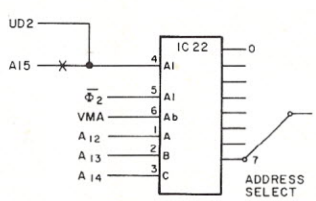


Fig. 2. MP-M.

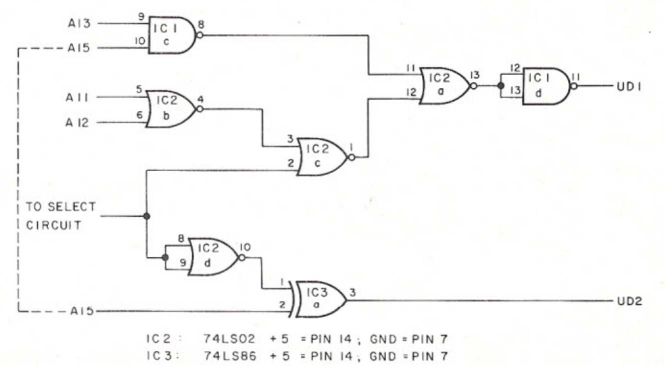


Fig. 3. New address decoding.

tines are still usable and where they belong. There is no need to change an addresses for software currently in use.

A Final Note:

Do not power-on the computer with the RAM board set at \$F000. The processor will rapidly jump to never-never land until brought back by a reset with MIKBUG set correctly. ■

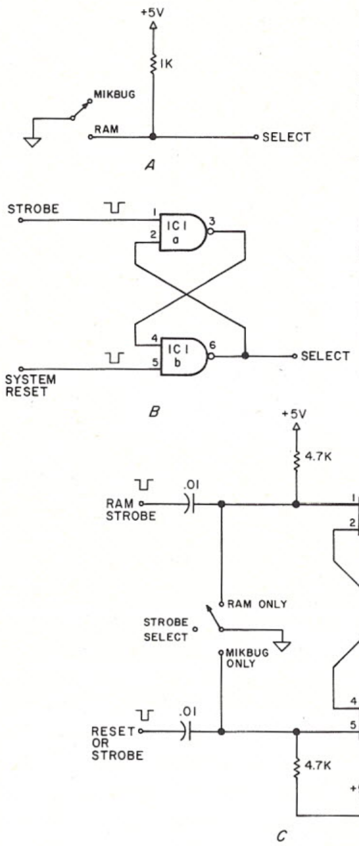


Fig. 4. Select circuits.

X-Y Plotting With an X-T Plotter

Produce publishable plots with a cheapo chart recorder.

```
2000 REM SHELL-METZNER SORT ON COL K0
2010 N=99:M=N
2020 M=INT(M/2)
2030 IF M=0 THEN 2170
2040 K=N-M:J=1
2050 I=J
2060 L=I+M:IF M%(I+2,K0)>M%(L+2,K0) THEN 2090
2070 J=J+1:IF J>K THEN 2020
2080 GOTO 2050
2090 REM SWAP ROWS OF M%
2100 FOR P=0 TO 3
2110 M%(0,P)=M%(I+2,P)
2120 M%(I+2,P)=M%(L+2,P)
2130 M%(L+2,P)=M%(0,P)
2140 NEXT P
2150 I=I-M:IF I<1 THEN 2070
2160 GOTO 2060
2170 PRINT "SORTED ON COL";K0
2180 RETURN
```

Program for Shell-Metzner sort of rows 2-101 of matrix M% of column K0. Rows 0 and 1 of the matrix contain identification, not data; row 0 is used for temporary storage during the sort. Entire rows are swapped in the loop at 2100-2140. A total of 99 rows of data is ordered on the desired column in about two minutes of computer time.

```
3500 REM PLOT PT SSR
3505 IF K<0 THEN K=0
3510 G%=INT(K*SC)+128
3514 IF G%>383 THEN G%=127
3515 IF G%>255 THEN G%=G%-256
3530 OUT 30,G%:GOSUB 2500
3540 IF KT<TI*32 THEN 3530
3550 RETURN
```

*Program to plot a point Y at a specific time T=X. Here the value (Y) to be plotted is K. SC is a scale factor, adjusted elsewhere in the program to ensure that all points fit on the page. Statements 3510-3515 adjust the value to agree with the convention used by the Cromemco D + 7A board; this is sent by the OUT 30, G% in 3530. Subroutine 2500 reads the clock and returns a count KT; this is compared to the current value of X, variable TI. The time scale factor 32 is adjusted to suit the paper speed being used. The value of Y is repeatedly sent to the plotter until KT=TI*32, then the RETURN is executed and the next point is processed.*

Kenneth Reid
1935 Trevilian Way
Louisville, KY 40205

An X-Y plotter capable of the precision needed to produce publishable plots is expensive—typically over $1000. A chart recorder of equivalent capability, the Heathkit model IR-18M, costs less than $200, but can only plot a variable Y against time. A Y-T plotter can be used as an X-Y plotter with a microcomputer. The trick is to convert the independent variable X to time.

How to Do It

I use an Altair 8800 with 24K of memory. The programs are written in Altair (Microsoft) BASIC, version 3.3. Analog data is collected using a Cromemco D+7A converter board (see *Microcomputing*, March 1979, p. 40) and saved in Tarbell format on cassette tape. An independent clock that counts seconds is available from a home-brew board.

Conversion of X to T occurs in two stages. The first is software: the data array is read in from tape as a 101 row × 4 column integer matrix and sorted on the column selected as variable X so that the values of this column are in ascending order. At the same time the other columns are sorted so that the rows are rearranged, but each row remains intact. Using a Shell-Metzner sort, this operation takes about two minutes on my system. (See "Quicksort," *Microcomputing*, April 1979, p. 96.)

The second step is the actual plotting. After plotting a set of calibration marks, the seconds clock resets to zero and begins to count. When the clock count equals or exceeds the first value of column X, the value in the same row for variable Y goes to the plotter pen via the D/A converter. The next X value is then compared, and its Y value sent at the time T exceeds X. This procedure is repeated until the time count T exceeds the largest value of X and the plot is completed (see Fig. 1.)

A useful variation is the histogram format of Fig. 2. For a specified X interval, the corresponding values of Y are averaged, and the mean value is plotted as a bar. The standard deviation is shown as a line above and below the bar.

The procedure is slow, particularly for the histogram format. Since the paper cannot move appreciably during the tracing of a vertical line, and the pen slew time is about one second for full scale, I'm limited to a paper speed of 50 sec/inch or slower. Usually I plot at 100 sec/inch or, for compact plots, 200 sec/inch. One advantage of the Y-T format is that the paper comes

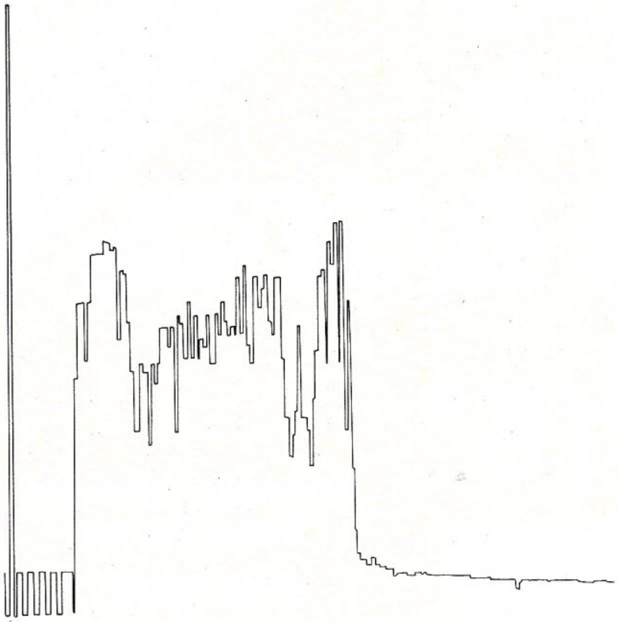


Fig. 1. Sample X-Y plot. Note that the intervals between direction changes of the plot line are variable, due to the X-T conversion. The tall bar at the left defines the available plotting height for Y, while the series of square waves specifies the scaling factor used, and the paper speed.

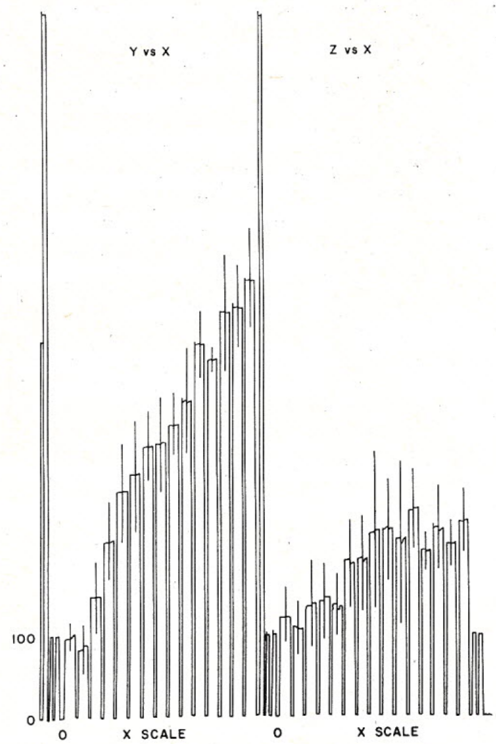
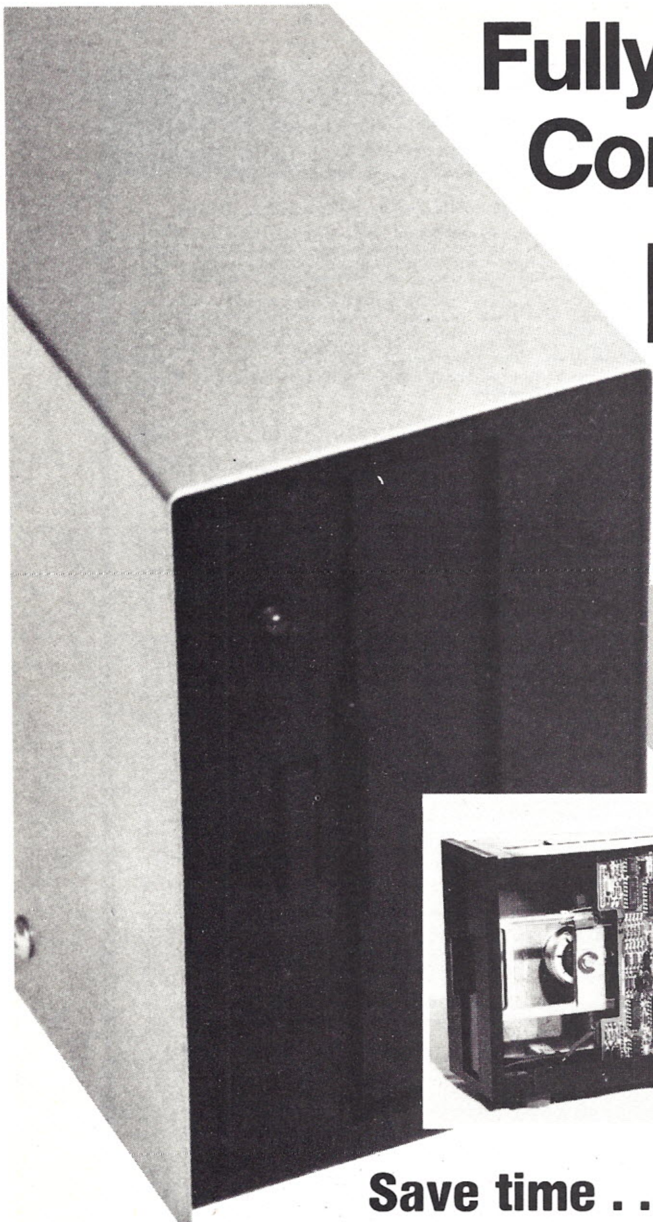


Fig. 2. Pair of plots showing the histogram mode. Two dependent variables, Y and Z, are plotted against X. The value of X ranges from 0 to 700; the bar width is 50. For the first bar, all Y values for rows in which X was between 0 and 50 were averaged, and the result plotted as the height of the bar. The thin line at the midpoint of the bar indicates the standard deviation of the mean for this group of Y values. If no standard deviation is shown, there is only one data point in the interval.

in a roll, letting me run a whole series of plots unattended. This is more convenient than using an X-Y plotter in which a new sheet of paper must be inserted by hand for each plot.■



Fully TRS-80® Compatible

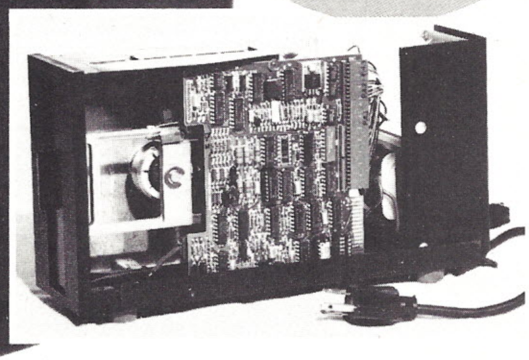
MPI B/51 DISK DRIVE

FREE CATALOG
Hundreds of other computer products at major savings. Cut out this coupon and send with your name and address.

$315[95]

plus $5 per drive shipping

In Stock! Limited Quantities.



Includes: Case and Power Supply

Fully tested
Guaranteed for 90 days!

Here's why the MPI/B-51 is the drive for success:

- 40 tracks
- 5 ms track-to-track
- Auto-eject
- Hi-Temp stability
- Fully-closable door
- Speed constant <1½%
- Switchable 220V/50kz or 110V/60kz
- Optical sensors—no switches
- 102K per disk

ADDS MORE POWER TO YOUR SYSTEM

Save time . . . Order by phone
Toll free: 1-800-323-4335
IN ILLINOIS CALL: 312-251-5955

TRS-80™ TANDY CORP.

150

Other Money Savings Opportunities Order by Phone or Mail

Cables
2 drive $34.95
4 drive 44.95

Diskettes
Verbatim .... 10 for $29.95
Memorex .... 10 for $26.95
Georgia Magnetics ... 10 for $28.95
Dysan .... 5 for $24.95
Plastic file box .... $3.95 each

Operating Systems
TRS DOS 2.2 .... $14.95
Percom Patch #4 ...... $9.95
New DOS 40 track .. $49.95
New DOS+ 40 track .. $99.95
TRS DOS manual $5.95

TRS-80
16K level II with keypad $749.95
Expansion interface 0K $269.95

Printers
Centronics 779-2 .. $1,050.00
Centronics 730 ...... $899.00
IDS-440 .... $949.00
NEW! TX-80.......... $945.00
NEC Spinwriter 5530 .... $2,679.00

16K Memory Kits
300ns $69.95
250ns $74.95
200ns $84.95
MPI Service Manual $3.00
MPI Engineering Manual $30.00

MIDWEST COMPUTER PERIPHERALS
P.O. BOX 437 • WILMETTE, ILLINOIS 60091

Midwest Computer Peripherals
P.O. Box 437 • Wilmette, Illinois 60091

ACCEPTED!

085

Quantity	Description	$ each	Total
		6% Ill. Tax	
		Shipping ($2.50 min.)	
		TOTAL	

(Min. Order $10.00)

☐ Check enclosed
Bill my ☐ Visa
☐ Master Charge
Acc. No. ____________ Exp. ______
☐ Please send catalog

Name ____________
Address ____________
City ____________ State ______ Zip ______

Patching the SSB Assembler

Making a "super assembler" really super.

John L. Alford
10000 Midlothian Tpk., Box 11
Richmond, VA 23235

One of the three major disk systems for the SS-50 bus is provided by Smoke Signal Broadcasting. Although its random DOS is a blessing, and the system utilities and software provided with the system put it at the top of the heap, one shortcoming is the assembler utility available for it. Actually a modified Technical Systems Consultants package, it is a bare-bones program that can present problems and irritations if it is used for more than relatively short assemblies.

Line Numbers

My first problem with the Smoke Signal assembler was line numbers: there weren't any. It wasn't a problem at first. A quick routine to increment a double-precision packed BCD number and display it did the trick. I had line numbers . . . all over the place! On every line, as well as the top, bottom, edges and corners of every page, line numbers appeared. I quickly learned what too much of a good thing meant. That's what happens when you're in a hurry.

The first necessity for numbering is a way to increment the count, but only on source lines, not expansion lines generated by the assembler, such as FCBs, FCCs and SPCs. Look at the program listing. BUMPLI, from lines 122 through 137 in the program listing, accomplishes this by being called from a point in pass two that jumps to parse each line before further assembly. Lines 50–51 are the assembler patch, and BUMPLI exits by jumping to the parse routine called by the original code. The string LINCNT, which BUMPLI increments, is the actual line number that is later optionally printed.

LINCNT must be output only on selected lines. For clarity, I wanted comment lines unnumbered, so I had to find a patch point to allow this. Lines 70–71 patch the number output routine OUTLIN into the listing formatting routine, which Smoke Signal appended to the original assembler to columnize the printout from free-form source code.

Referring to OUTLIN (lines 110–118), the list format routine picks up the first character in the label column. If it is an asterisk, the line is a comment. OUTLIN simply performs the test the assembler used to do the kill column formatting on comments, and exits back to the old code to handle them. After inserting line numbers in the desired lines, OUTLIN jumps back into the old code to handle the rest of the line.

A note here is in order: the patches shown do not take into account the possibility of a system using a 40-column printer such as the SWTP PR-40. The column numbering in the original code has not been patched to take into account the inclusion of six more columns. If you use the PR-40, some experimenting is in order.

Other than that, the assembler couldn't have been more congenial to these patches. Scan the listing and notice that no provision is made to eliminate printing line numbers on skipped lines and expansion lines. The reason is that the assembler doesn't use the LIST routine on those lines, so OUTLIN never is called. One problem down, and one to go.

Optional Printing

For long listings, the option of only printing the part of a listing damaged by a hungry printer is a must! The patches I have written provide for that. Fig. 1a shows a typical DOS call for assembly. Commas are delimiters. The source file ASPA10.SRC should be assembled, generating no object file, with a listing to be printed (as opposed to being displayed) without a symbol table listing.

After employing the patches in the listing, the line looks like Fig. 1b. The only change is that now only pages one through four are printed. Fig. 1c is a similar call to the patched assembler, but the option is set to print all pages from page one to the end. Fig. 1d requests that all pages up to and including page four be printed. Fig. 2 shows the format of the new call.

Refer to the listing again. Before the pass-one initialization occurs, the page options are picked up from the DOS line buffer. The SETLIM routine from lines 141–173 accomplishes this. Lines 80–81 are the patch that calls SETLIM, which exits by performing the jump to the pass-one initialization replaced by the patch.

If an error is detected during SETLIM, the limits default to as many of the options as were picked up before the error. This means that if the format in Fig. 2 is not followed, the lower limit may be properly set, but the upper limit will default to the end of the listing.

The second half of the option routine does several things. PROUT (lines 177–218) deter-

```
A -   ASMB,ASPA10.SRC,,P,,N
B -   ASMB,ASPA10.SRC,,P,,N,01,04
C -   ASMB,ASPA10.SRC,,P,,N,01
D -   ASMB,ASPA10.SRC,,P,,N,N,04
```

Fig. 1. DOS Assembly calls.

```
PRINT OPTION                                  LIST OPTION
 OBJECT FILE                                  SYMBOL OPTION
 SOURCE FILE

          (IFILE),(OFILE),(P),(N),(N),(XX),(XX)

 FIRST PAGE OPTION *
  LAST PAGE OPTION *

* NOTE: IF XX IS NOT TO BE SPECIFIED 'N' MAY BE SUBSTITUTED
        OR, IF OPTION LINES ARE  NOT USED, ',XX,XX' MAY BE
        DROPPED.  IF, HOWEVER, A PAGE OPTION IS TO BE USED
        A TWO-DIGIT NUMBER MUST BE USED EVEN IF ZERO (00).
```

Fig. 2. New assembly call format.

mines whether to print a page (lines 206–211), keeps up with the location on a page (lines 188–197), handles the printer/display decision (lines 214–218) and allows form-feed characters generated by the assembler (ASCII $0C) to cause form-feed on printers that don't support top-of-forms (lines 179–185).

PROUT is called from the jump addresses patched in lines 62–65. With the patches shown in the listing in place, change the addresses in lines 217–218 for display and output routines instead of the old addresses called for by the instruction manual. Lines 60–61 replace the form-feed string incorporated in the original assembler. The new string, together with PROUT, causes an eject to top-of-form and prints a period at the tear point to allow a check on alignment for continuous forms.

Lines 53–57 set up the assembler to handle a 66-line page (8 1/2 × 11 inch forms at six lines to the inch). Notice that the counter location for PROUT, LINECT initially has a value of three because the assembler doesn't know if the listing will run under the PAG option until the OPT pseudo is parsed. Assuming the assembler OPT pseudo is on line two, directly after NAM, two lines print before the assembler starts counting lines (see Listing 1).

One Last Patch

Invariably, in assembly-language programming, you have a stack of listings with changes and updates, and you have no idea which listing is the most current. To overcome this problem, I included the routine MOVSRC (lines 88–106).

MOVSRC takes the source filename from the read-file control block and stuffs it into the title line in place of SSB MNEMONIC ASSEMBLER. Lines 75–77 patch the banner, and lines 73–74 patch MOVSRC into the end of pass one. Obviously, the program listing was generated from ASPATS.X12. If any pages are replaced later using the print page option, the updated filename will appear on those pages only. This provides numerous benefits.

Conclusion

The patches presented in this article provide one way to make the Smoke Signal assembler as flexible as practical without going to high-priced macro-assemblers. As a convenience for those who wish to incorporate these changes into the SSB assembler but don't like to type, or who wish to make changes without reentering the entire source file, a disk is available from the author for $17.95, shipping and handling included. Included on the disk is a copy of the source file and assembled object code plus any updates incorporated after publication. ■

```
0001:       NAM    ASSEMBLER OPTIONAL PRINT OVERLAY
0002:       OPT    NOG,PAG
```

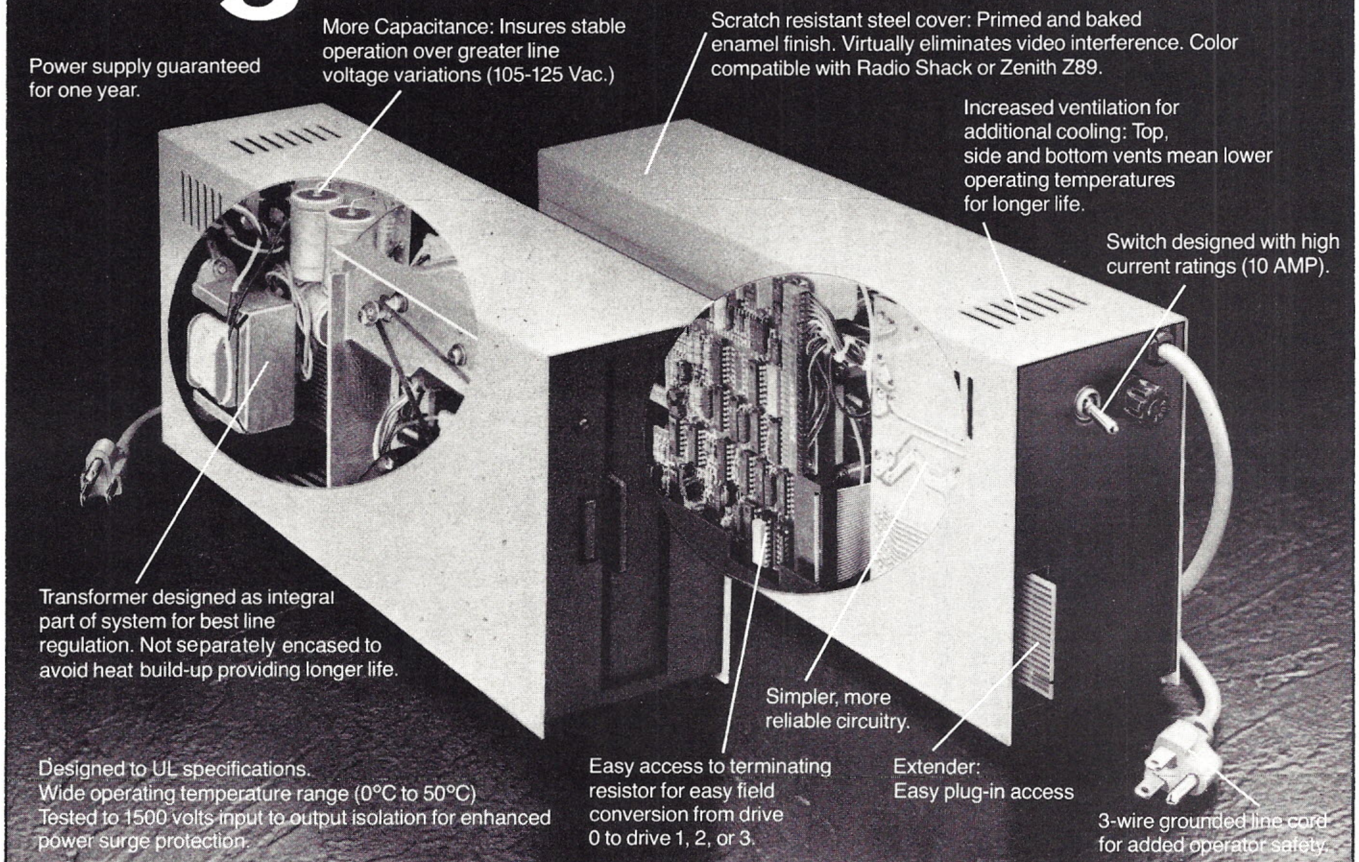
Listing 1.

Program listing.

```
                   * GENERAL PROGRAM EQUATES

0038               0029: PRNFLG EQU    $0038       PRINTER OUTPUT FLAG
0326               0030: P1INIT EQU    $0326       PASS ONE INITIALIZE
03D9               0031: PASTWO EQU    $03D9       ENTRY POINT FOR PASS TWO
07AB               0032: PDATA  EQU    $07AB       OUTPUT A STRING
0B75               0033: PARSE  EQU    $0B75       LINE PARSING ROUTINE
11B1               0034: TITLES EQU    $11B1       POINTER FOR SOURCE NAME
162F               0035: INCH   EQU    $162F       KEYBOARD INPUT
1638               0036: COUCH  EQU    $1638       TERMINAL OUTPUT
171B               0037: GETIT  EQU    $171B       GET NEXT CHAR FROM IBF
1798               0038: DASOUC EQU    $1798       PRINTER/TERMINAL OUTPUT
17A3               0039: RFCB   EQU    $17A3       INPUT FILE CONTROL BLOCK
19C4               0040: LIST1  EQU    $19C4       RE-ENTRY POINT TO LIST
1A81               0041: COMMT  EQU    $1A81       COMMENT HANDLER
729A               0042: ZANCHK EQU    $729A       CHECK FOR ALPAHNUMERIC
E1D1               0043: OUTEEE EQU    $E1D1       SCREEN OUTPUT
E5B1               0044: PRINTR EQU    $E5B1       PRINTER HANDLER OUTPUT
E59D               0045: PRINIT EQU    $E59D       MY PRINTER INITIALIZATION

                   * ASSEMBLER PROGRAM PATCHES

                   * PASS TWO PATCH TO INCREMENT LINE COUNT
03E8               0050:        ORG    $03E8
03E8 BD 1A EC      0051:        JSR    BUMPLI
                   * PATCH FOR LINES OF PRINT PER PAGE
07C5               0053:        ORG    $07C5
07C5 3A            0054:        FCB    $3A
                   * PATCH TO SET LINES FROM TOF TO TITLE
1143               0056:        ORG    $1143
1143 05            0057:        FCB    5
                   * PATCH FOR TOP OF FORM STRING
11D1               0059:        ORG    $11D1
11D1 0C            0060:        FCB    $C,$2E,$4
                   * PATCH FOR FORMAT HANDLER
1632               0062:        ORG    $1632
1632 7E 1B 61      0063:        JMP    PROUT
1635 7E E5 9D      0064:        JMP    PRINIT
1638 7E 1B 61      0065:        JMP    PROUT
                   * PATCH TO SET NEW END OF PROGRAM
163B               0067:        ORG    $163B
163B CE 1B DE      0068:        LDX    #NEWEND
                   * PATCH TO OUTPUT LINE NUMBERS
19BD               0070:        ORG    $19BD
19BD 7E 1A D4      0071:        JMP    OUTLIN
                   * PATCH TO INSERT SOURCE FILE NAME INTO BANNER
```

```
1711              0073:        ORG    $1711
1711 BD 1A A7     0074:        JSR    MOVSRC
                  * NEW TITLE LINE
11B1              0076:        ORG    $11B1
11B1 46           0077:        FCC    'FILENAME:                      PAGE '
11D0 04           0078:        FCB    4
                  * PATCH TO PICK UP PAGE LIMITS FROM COMMAND
16DB              0080:        ORG    $16DB
16DB BD 1B 0D     0081:        JSR    SETLIM

                  * START OF NEW ROUTINES
1AA7              0084:        ORG    $1AA7

                  * MOVE FILE NAME FROM FCB TO TITLE AREA

1AA7 CE 11 C4     0088: MOVSRC LDX    #TITLES+19 POINT INTO TITLE STRING
1AAA FF 1B D0     0089:        STX    TEMPX2     AND SAVE IT
1AAD CE 17 AE     0090:        LDX    #RFCB+11   POINT TO FCB
1AB0 A6 00        0091: MOVSR1 LDA A  X          GET LAST CHARACTER
1AB2 26 02        0092:        BNE    *+4        BRANCH IF ALPHANUMERIC
1AB4 86 20        0093:        LDA A  #$20       OTHERWISE PAD IT
1AB6 09           0094:        DEX               BUMP POINTER
1AB7 FF 1B CE     0095:        STX    TEMPX1     SAVE IT
1ABA FE 1B D0     0096:        LDX    TEMPX2     GET NAME POINTER
1ABD A7 00        0097:        STA A  X          SAVE IT
1ABF 09           0098:        DEX               BUMP POINTER
1AC0 8C 11 C1     0099:        CPX    #TITLES+16 EXTENSION POINT?
1AC3 26 01        0100:        BNE    MOVSR2     BRANCH IF NOT
1AC5 09           0101:        DEX               BUMP POINTER ONCE MORE
1AC6 FF 1B D0     0102: MOVSR2 STX    TEMPX2     SAVE POINTER
1AC9 FE 1B CE     0103:        LDX    TEMPX1     RESTORE FROM-POINTER
1ACC 8C 17 A5     0104:        CPX    #RFCB+2    TRANSFER COMPLETE?
1ACF 26 DF        0105:        BNE    MOVSR1     LOOP UNTIL DONE
1AD1 7E 03 D9     0106:        JMP    PASTWO     GO HANDLE PASS TWO

                  * OUTPUT LINE NUMBER ON SELECTED LINES

1AD4 81 2A        0110: OUTLIN CMP A  #$2A       COMMENT LINE?
1AD6 27 11        0111:        BEQ    OUTLIX     IF SO, EXIT
1AD8 FF 1B CE     0112:        STX    TEMPX1     SAVE LINE POINTER
1ADB CE 1B D7     0113:        LDX    #LINCNT    POINT TO LINE STRING
1ADE BD 07 AB     0114:        JSR    PDATA      OUTPUT THE STRING
1AE1 FE 1B CE     0115:        LDX    TEMPX1     RESTORE X-REG
1AE4 A6 00        0116:        LDA A  X          RESTORE A-REG
1AE6 7E 19 C4     0117:        JMP    LIST1      RE-ENTRY TO LIST LINE
1AE9 7E 1A 81     0118: OUTLIX JMP    COMMT      RE-ENTRY TO HANDLE COMMENTS

                  * BUMP LINE NUMBER BEFORE PARSING

1AEC FF 1B CE     0122: BUMPLI STX    TEMPX1     SAVE REGISTERS
1AEF 36           0123:        PSH A
1AF0 CE 1B DB     0124:        LDX    #LINCNT+4  POINT TO STRING AGAIN
1AF3 09           0125: BUMPL1 DEX               BUMP POINTER
1AF4 A6 00        0126:        LDA A  X          GET UNITS
1AF6 4C           0127:        INC A             BUMP IT
1AF7 A7 00        0128:        STA A  X          SAVE IT
1AF9 81 3A        0129:        CMP A  #$3A       OVERFLOW?
1AFB 26 09        0130:        BNE    BUMPLX     IF NOT, EXIT
1AFD 86 30        0131:        LDA A  #$30       OTHERWISE GET ZERO
1AFF A7 00        0132:        STA A  X
1B01 8C 1B D7     0133:        CPX    #LINCNT    DONE?
1B04 26 ED        0134:        BNE    BUMPL1     IF NOT, LOOP
1B06 32           0135: BUMPLX PUL A             RESTORE REGISTERS
1B07 FE 1B CE     0136:        LDX    TEMPX1
1B0A 7E 0B 75     0137:        JMP    PARSE      GO PARSE LINE

                  * SET PRINTER LIMITS

1B0D 81 0D        0141: SETLIM CMP A  #$D        NO LIMIT OPTION?
1B0F 27 4D        0142:        BEQ    SETLIX     EXIT IF NONE
1B11 BD 17 1B     0143:        JSR    GETIT      GET NEXT CHAR
1B14 27 48        0144:        BEQ    SETLIX     EXIT IF CR
1B16 BD 72 9A     0145:        JSR    ZANCHK     CHECK IF ALPHANUMERIC
1B19 24 05        0146:        BCC    SETLI0     BRANCH IF OPTION NUMBER
1B1B BD 17 1B     0147:        JSR    GETIT      GET FIRST OPTION NUMBER
1B1E 27 3E        0148:        BEQ    SETLIX     EXIT IF NONE
1B20 81 4E        0149: SETLI0 CMP A  #'N        NO LOWER LIMIT?
1B22 27 16        0150:        BEQ    SETLI1     BRANCH IF NOT
1B24 BD 1B B5     0151:        JSR    NUMBER     CHECK FOR GOOD NUMBER
1B27 25 35        0152:        BCS    SETLIX     EXIT IF NOT
1B29 B7 1B D2     0153:        STA A  TEMPA      SAVE IT
1B2C BD 17 1B     0154:        JSR    GETIT      GET NEXT
1B2F BD 1B B5     0155:        JSR    NUMBER     CHECK FOR GOOD
1B32 25 2A        0156:        BCS    SETLIX     EXIT IF NOT
1B34 BD 1B C2     0157:        JSR    ADDNUM     ADD HALVES OF BYTE
1B37 B7 1B D4     0158:        STA A  LOWLIM     AND SET LOWER LIMIT
1B3A BD 17 1B     0159: SETLI1 JSR    GETIT      GET NEXT
1B3D BD 17 1B     0160:        JSR    GETIT
1B40 27 1C        0161:        BEQ    SETLIX     EXIT ON NO UPPER
1B42 81 4E        0162:        CMP A  #'N        CHECK FOR NO COMMAND
1B44 27 18        0163:        BEQ    SETLIX     EXIT IF SO
1B46 BD 1B B5     0164:        JSR    NUMBER     GOOD ENTRY?
1B49 25 13        0165:        BCS    SETLIX     EXIT IF NOT
1B4B B7 1B D2     0166:        STA A  TEMPA      SAVE IT
```

Precision Engineered Drives...



With the number of disk drives on the market increasing, more and more people are beginning to ask what's underneath that cover.

The CCI™ series of disk drives have been designed for long life and ease of operation. The features shown above are what set our CCI drives apart from the rest. With a CCI drive you get an integrated professional design!

If you're still not convinced that you get the most for your money with a CCI drive, just ask for our complete specifications sheet. Then, compare our disk drives to anyone else's.

5 1/4" DRIVES

CCI-100	40 Track (102K Bytes) for TRS-80* Model I	**$399.00**
CCI-189	40 Track (102K Bytes) for Zenith Z89	**$499.00**
CCI-200	77 Track (197K Bytes) for TRS-80* Model I	**$675.00**

8" DRIVES

CCI-800	77 Track (1/2 Meg Bytes) for TRS-80* Model II	**$895.00**

All CCI drives are also available for 220 Vac (50Hz) operation.

Operating Systems

NEWDOS Plus for 5 1/4", 40 and 77 Track Drives—with over 200 modifications and corrections to TRSDOS **$110.00**
CP/M for Model I, Zenith **$150.00**
CP/M for Model II, Altos **$199.00**

Software by S&M Systems

INSEQ-80™—Indexed Sequential Access Method (**ISAM**) for the TRS-80 Model I.

Four machine language programs that can be called from your BASIC program via USR functions to access records either sequentially or randomly. The INSEQ-80 programs maintain all indexes and chains for you. Includes reorganization utility to consolidate files. **$49.95**

Professional Business Software using INSEQ-80 for the TRS-80* Model I and Zenith Z89.

Accounts Payable, Accounts Receivable, General Ledger, Payroll per package **$99.00**
Inventory per package **$125.00**

ComputerCity™

175 Main Street, Dept. K-7, Charlestown, MA 02129
Hours: 10AM-6PM (EST) Mon.-Fri. (Sat. till 5).

TO ORDER CALL
TOLL FREE 1-800-343-6522
TWX: 710-348-1796
Massachusetts residents call 617/242-3350
For detailed technical information, call 617/242-3350.
Freight Collect, F.O.B. Charlestown.
*TRS-80 is a trademark of the Tandy Corporation

Products also available from: Radio Shack, NEC, Centronics, Paper Tiger, TI, Altos, MPI, Zenith, ATARI, Mattel, PET, OKIDATA, Apple, Eaton/LRC.

FRANCHISE AND DEALER (NATIONAL/INTERNATIONAL) INQUIRIES INVITED ✓278

Retail Stores: MA: Burlington, Charlestown, Framingham, Hanover **NH:** Manchester **RI:** Providence

master charge THE INTERBANK CARD
VISA

```
1B4E BD 17 1B            JSR    GETIT     GET NEXT NIBBLE
1B51 27 0B               BEQ    SETLIX    EXIT ON ERROR
1B53 BD 1B B5            JSR    NUMBER    CHECK IT
1B56 25 06               BCS    SETLIX    ON NON-NUMERIC
1B58 BD 1B C2            JSR    ADDNUM    OTHERWISE ADD NIBBLES
1B5B B7 1B D3            STA A  HILIM     SAVE HI LIMIT
1B5E 7E 03 26     SETLIX JMP    P1INIT    AND GO ASSEMBLE

                  * OUTPUT CHARACTER TO PRINTER AND HANDLE PAGING

1B61 81 0D        PROUT  CMP A  #$D       CARRIAGE RETURN?
1B63 27 12               BEQ    OUTCR     IF SO, DO IT
1B65 81 0C               CMP A  #$C       FORM FEED?
1B67 26 30               BNE    OUTCH     IF NOT, OUTPUT CHARACTER
1B69 8D 08        PROUT1 BSR    OUTCR0    OUTPUT CR-LF
1B6B B6 1B D6            LDA A  LINECT    GET LINE COUNTER
1B6E 81 01               CMP A  #1        TOP OF FORM?
1B70 26 F7               BNE    PROUT1    LOOP UNTIL IT IS
1B72 39                  RTS              THEN RETURN
1B73 86 0A        OUTCR0 LDA A  #$A       GET A LINE-FEED
1B75 8D 22               BSR    OUTCH     AND OUTPUT IT
1B77 B6 1B D6     OUTCR  LDA A  LINECT    GET LINE COUNT
1B7A 4C                  INC A            BUMP IT
1B7B 81 44               CMP A  #$44      END OF FORM?
1B7D 26 0B               BNE    OUTCR1    BRANCH IF NOT
1B7F B6 1B D5            LDA A  PAGCNT    OTHERWISE GET PAGE COUNT
1B82 8B 01               ADD A  #1        BUMP IT
1B84 19                  DAA              RESTORE TO BCD
1B85 B7 1B D5            STA A  PAGCNT    AND SAVE IT
1B88 86 01               LDA A  #1        AND SET TO TOP OF FORM
1B8A B7 1B D6     OUTCR1 STA A  LINECT    AND SAVE COUNT
1B8D 37                  PSH B            SAVE B-REG
1B8E C6 04               LDA B  #4        GET PAD NULL COUNT
1B90 86 0D               LDA A  #$D       GET A CR
1B92 8D 05               BSR    OUTCH
1B94 4F                  CLR A            AND SET FOR PADDING NULLS
1B95 5A                  DEC B            BUMP PAD COUNT
1B96 26 FA               BNE    *-4       AND LOOP UNTIL DONE
1B98 33                  PUL B            RESTORE B-REG
1B99 36           OUTCH  PSH A            SAVE CHARACTER
1B9A B6 1B D5            LDA A  PAGCNT    GET CURRENT PAGE
1B9D B1 1B D4            CMP A  LOWLIM    CHECK RANGE
1BA0 25 05               BCS    OUTCH0    BRANCH IF NOT TIME
1BA2 B1 1B D3            CMP A  HILIM     CHECK RANGE
1BA5 23 02               BLS    OUTCH1    BRANCH IF NOT DONE
1BA7 32           OUTCH0 PUL A            GET CHARACTER
1BA8 39                  RTS              RETURN IF NOT IN RANGE
1BA9 32           OUTCH1 PUL A            RESTORE CHARACTER
1BAA 7D 00 38            TST    PRNFLG    CHECK IF PRINTER
1BAD 26 03               BNE    OUTCH2    BRANCH IF SO
1BAF 7E E1 D1            JMP    OUTEEE    OUTPUT TO SCREEN
1BB2 7E E5 B1     OUTCH2 JMP    PRINTR    GO OUTPUT TO PRINTER

                  * CHECK FOR ASCII NUMERIC

1BB5 81 30        NUMBER CMP A  #$30      CHECK FOR NUMERIC
1BB7 25 05               BCS    NUMBEX    EXIT IF NOT
1BB9 81 39               CMP A  #$39      NEXT CHECK
1BBB 23 02               BLS    NUMBE1    BRANCH IF OK
1BBD 0D                  SEC              SET CARRY
1BBE 39           NUMBEX RTS              AND EXIT
1BBF 0C           NUMBE1 CLC              CLEAR CARRY
1BC0 20 FC               BRA    NUMBEX    AND EXIT

                  * ADD ACCUMULATOR TO TEMPA

1BC2 84 0F        ADDNUM AND A  #$F       MASK TO BCD
1BC4 16                  TAB              SAVE IT
1BC5 B6 1B D2            LDA A  TEMPA     GET OTHER HALF
1BC8 48                  ASL A            AND SHIFT INTO HI NIBBLE
1BC9 48                  ASL A
1BCA 48                  ASL A
1BCB 48                  ASL A
1BCC 1B                  ABA              ADD IN LOW
1BCD 39                  RTS              AND RETURN

                  * TEMPORARY STORAGE

1BCE 00 00        TEMPX1 FDB    0
1BD0 00 00        TEMPX2 FDB    0
1BD2 00           TEMPA  FCB    0
1BD3 99           HILIM  FCB    $99
1BD4 00           LOWLIM FCB    0
1BD5 00           PAGCNT FCB    0
1BD6 03           LINECT FCB    3         ADJUSTED FOR TITLE

                  * MESSAGE STORAGE

1BD7 30           LINCNT FCC    '0000: '
1BDD 04                  FCB    4
1BDE              NEWEND EQU    *
                         END
```

Disassembler For the 1802

More efficient and readable than coding sheets is this 1802 disassembler.

John Beringer
2728 West Sahara, Apt. 2
Las Vegas, NV 89102

By the time I've gone from the beginning to the end of an 1802 program, written for my Elf II, I have accumulated many sheets of code (not necessarily arranged in any particular order). If I were to reexamine these numerous sheets (which I rarely do), I would find on them scratchings, scribblings, calculations and patches written in the narrow margins complete with arrows pointing to the address of insertion in the once-neat column of op codes. Perhaps this is why I do not often reread any finished work.

Oh, but wait! There was a certain hexadecimal-to-base-ten conversion routine that worked especially well in this or that program. Now, where was it? After searching twice through 23 pages of unreadable code, I finally find the routine. Now to the user's manual to see what the op codes mean. Wonder why I did it that way? Oh, no! Long Branch to the end of the program. Back to the stack to look for page 23 to see what happens there, and so on.

What I need is a neat, attractive way of displaying what I have written. Hence, the Disassembler. And since software for the 1802 has been as scarce as diamonds in dirt, I'll have to write it myself. First, I need a character generator. Add a few tables of op code mnemonics, stir well with a dash of patience, and the job is done.

Program Breakdown

Addresses 0000 through 0014 contain an initialization routine that sets the interrupt pointer, R1, to 0017 (the TV chip display routine); R2 (stack pointer) is initialized to 06FF (top of push-down stack); R3 points to 0399, the beginning of the program. R3 becomes the program counter with an SEP instruction.

The subroutines occupy two areas of memory, beginning and end, while the main program occupies the middle of used memory. The reason for this is that OUTCH, the character generator routine, was orig-

0000-0014	Initialization routine.
0015-0032	Display routine.
0033-00FF	Part of OUTCH subroutine. Outputs a single character to screen.
0100-01FF	Screen display buffer.
0200-02B3	Table of character patterns.
02B4-0398	Rest of OUTCH.
0399-0644	Disassembler Main. Displays mnemonics of 1802 op codes.
0645-064F	GETHEX subroutine. Inputs and displays one byte from hex keypad.
0650-065B	OUTMSG subroutine. Outputs an entire character.
065C-0668	OUTDGT subroutine. Converts single hex digit to ASCII.
0669-06FF	Stack.
0700-071F	Table of 3N mnemonics.
0720-075F	Table of 7N mnemonics.
0760-078F	Table of CN mnemonics.
0790-07BF	Table of FN mnemonics.

Table 1. Breakdown of memory used by Disassembler.

Register	Use
0	DMA pointer for TV display chip.
1	Pointer to interrupt routine to service display chip.
2	Pointer to stack (push-down from 06FF).
3	Program counter.
4	Not used.
5	Not used.
6	Not used.
7	Pointer to 3N, 7N, CN, FN instruction mnemonics.
8	Pointer to OUTDGT.
9	Pointer to GETHEX.
A	Pointer to OUTCH.
B	Pointer to OUTMSG.
C	End address of segment to be disassembled.
D	Start address of segment to be disassembled.
E	Shift counter used by OUTCH (for spacing between characters).
F	Points to address of next screen character display area (used by OUTCH).

Table 2. Use of registers.

inally a stand-alone program capable of accepting and displaying input from an ASCII keyboard. The other subroutines evolved as the main program was being written. OUTCH resides in addresses 0033 through 0398, while GETHEX, OUTMSG and OUTDGT occupy addresses 0645 through 0668. The disassembler main program resides in memory 0399 through 0644. Some mnemonic tables occupy 0700 to 07BF, while other mnemonics needing special consideration are incorporated within the main program. Table 1 summarizes the memory breakdown.

Operating the Disassembler

Addresses 0000 through 0002 contain an LBR command to jump to the Elf monitor program. If another operating system is used, this instruction should be modified to contain the entry point address of that system.

After the program is loaded and the RUN switch is turned on, control is passed to the disassembler by jumping to address 0003. With the Elf II, this is accomplished by entering 00, 00, 03. Other systems may require different entries.

When the disassembler executes, it will display an input prompting message. Enter the desired starting address, high byte first, then low byte. As each digit is entered, it is simultaneously displayed on the hex display. When the correct byte is showing, press the input button to enter the byte. After the two-byte starting address is entered, key in the two-byte ending address in the same manner. Press the input button again to display the first line of code. Continue pressing the input button to display each subsequent line of data. When the ending address is reached, END will appear. Pressing the input button once more will bring back the first prompting message. From this point, another set of starting and ending addresses may be entered.

In all, this program occupies 2K RAM, addressed 0000 through 07BF. If this is unsatisfactory, unload the entire program to tape and reload to a higher location (with the Elf; other systems may not allow this). Of course, the table and subroutine pointers, as well as the Long Branch instructions, would need to be modified in this situation.

Table 1 is a breakdown of the memory occupied by the disassembler. Table 2 shows how each register is used. Listing 1 is the disassembler program. Listing 2 is the table of dot patterns used by OUTCH to generate characters. Listing 3 is the instruction mnemonics that are displayed on the screen.

This disassembler was designed for anyone with a minimum Elf with 4K RAM memory and TV or display monitor. If you have access to a hard-copy device, OUTCH may be modified or replaced by a routine to output to that device. In any case, I hope that this program proves useful to anyone who utilizes it. I find it an excellent debugging tool, easily more readable than my haphazard coding sheets. ■

Netronics has recently released their own assembler, as well as a disassembler. However, the Netronics version sells for $20, a price that I consider prohibitive. Their version requires that the user buy their video display board (or at least a TVT).

Listing 1. 1802 Disassembler.

```
0000  C0 F0 00          LBR   F000     Jump to Elf monitor.
0003  90        START   GHI   R0       Get current addr hi.
0004  B1                PHI   R1       Current addr into R1 hi.
0005  F8 17             LDI   "17"     R1 points to...
0007  A1                PLO   R1         TV chip routine.
0008  F8 0F             LDI   "06"     R2 points to stack (06FF).
000A  B2                PHI   R2       -
000B  F8 FF             LDI   "FF"     -
000D  A2                PLO   R2       -
000E  F8 03             LDI   "03"     R3 points to MAIN (0399).
0010  B3                PHI   R3       -
0011  F8 99             LDI   "99"     -
0013  A3                PLO   R3       -
0014  D3                SEP   R3       Jump to MAIN.
0015  72        RETURN  LDXA           Restore data register.
0016  70                RET            Return to caller.
0017  22        INTRPT  DEC   R2       Save data register.
0018  78                SAV            Save R(P), R(X).
0019  22                DEC   R2       -
001A  52                STR   R2       -
001B  C4                NOP            Delay to sync.
001C  C4                NOP            -
001D  C4                NOP            -
001E  F8 01             LDI   "01"     R0 points to display area.
0020  B0                PHI   R0       -
0021  F8 00             LDI   "00"     -
0023  A0                PLO   R0       -
0024  80        OUT8    GLO   R0       Output 8 display bytes.
0025  E2                SEX   R2       -
0026  E2                SEX   R2       -
0027  20                DEC   R0       -
0028  A0                PLO   R0       -
0029  E2                SEX   R2       -
002A  20                DEC   R0       -
002B  A0                PLO   R0       -
002C  E2                SEX   R2       -
002D  20                DEC   R0       -
002E  A0                PLO   R0       -
002F  3C 24             BN1   OUT8     If TV chip not done,
                                         branch to get 8 more.
0031  30 15             BR    RETURN   Else done, return.
0033  12        RTN2    INC   R2       Restore stack pointer.
0034  70                RET            Return to caller.
0035  AE        OUTCH   PLO   RE       Save character.
0036  E2                SEX   R2
0037  8E                GLO   RE       Get character.
0038  FF 0D             SMI   "0D"     Character = CR?
003A  3A 4E             BNZ   TESTLF   No, branch.
003C  F8 01             LDI   "01"     RF points to end of
003E  BF                PHI   RF         display area.
003F  F8 FF             LDI   "FF"     -
0041  AF                PLO   RF       -
0042  2F        BLANK   DEC   RF       Screen minus 1.
0043  F8 00             LDI   "00"     Store 8 spaces on screen.
0045  5F                STR   RF       -
0046  8F                GLO   RF       Done? Screen blank?
0047  3A 42             BNZ   BLANK    No, branch and continue.
0049  F8 09     SHIFT9  LDI   "09"     Yes, set shift counter...
004B  BE                PHI   RE         to maximum.
004C  30 33             BR    RTN2     Branch to return.
004E  8E        TESTLF  GLO   RE       Get character.
004F  FF 0A             SMI   "0A"     Character = LF?
0051  3A 72             BNZ   TEST20   No, branch.
0053  8F                GLO   RF       Get screen addr.
0054  FF C0             SMI   "C0"     Bottom of screen?
0056  C3 03 3A          LBDF  SCROLL   Yes, scroll up screen.
```

ADDR	DATA															
0200	02	52	24	22	14	00	00	00	22	66	17	27	22	00	10	42
0210	22	62	67	72	57	15	44	47	62	63	75	54	55	73	06	20
0220	02	57	71	52	22	52	00	01	56	11	54	41	55	02	27	25
0230	55	55	54	45	52	15	46	65	55	54	25	54	55	12	42	50
0240	02	02	22	20	22	27	07	02	52	62	56	62	23	40	40	11
0250	55	64	56	64	72	16	45	55	65	62	25	55	22	22	22	00
0260	00	07	74	40	22	52	20	04	52	41	71	54	51	02	27	20
0270	47	55	54	45	52	55	44	45	42	51	25	56	52	42	12	05
0280	02	02	21	30	14	00	40	40	27	76	16	24	22	44	10	42
0290	75	62	67	43	57	25	74	47	41	56	27	24	52	73	06	02
02A0	00	01	00	00	00	01	01	01	01	01	00	00	00	01	00	00
02B0	01	00	01	00												

Listing 2. Character pattern table for OUTCH subroutine.

ADDR	DATA															
0700	52	51	5A	50	31	32	33	34	20	4E	4E	4D	4E	4E	4E	4E
0710	20	20	20	5A	20	20	20	20	20	51	5A	00	31	32	33	34
0720	52	44	4C	53	41	53	52	53	53	4D	52	53	41	53	52	53
0730	45	49	44	54	44	44	53	4D	41	41	45	45	44	44	53	4D
0740	54	53	58	58	43	42	48	42	56	52	51	51	43	42	48	42
0750	20	20	41	44	20	20	52	20	20	4B	20	20	49	49	4C	49
0760	42	42	42	42	4E	53	53	53	53	42	42	42	53	53	53	53
0770	52	51	5A	50	4F	4E	4E	4D	4B	4E	4E	4D	49	4B	4B	50
0780	20	20	20	5A	50	51	5A	20	50	51	5A	00	45	51	5A	5A
0790	4C	4F	41	58	41	53	53	53	4C	4F	41	58	41	53	53	53
07A0	44	52	4E	4F	44	44	48	4D	44	52	4E	52	44	44	48	4D
07B0	58	20	44	52	44	20	52	20	49	49	49	49	49	49	4C	49

07C0 User space begins

Listing 3. Table of instruction mnemonics.

```
0059  8F              GLO   RF        Get screen addr.
005A  FA OF           ANI   "OF"      Mask upper bits.
005C  3A 68           BNZ   BEGINL    Beginning of line?
005E  8F              GLO   RF        RF point to next line.
005F  FC 08           ADI   "08"      -
0061  AF              PLO   RF        -
0062  8F      NEWLN   GLO   RF        -
0063  FC  28          ADI   "28"      -
0065  AF              PLO   RF        -
0066  30 49           BR    SHIFT9    Branch to set shift, rtn.
0068  1F      BEGINL  INC   RF        Line addr plus 1.
0069  8F              GLO   RF        Get screen addr.
006A  FA OF           ANI   "OF"      Mask upper bits off.
006C  FF 08           SMI   "08"      Test addr. Begin a line?
006E  32 62           BZ    NEWLN     Yes, branch.
0070  30 68           BR    BEGINL    No, loop and continue.
0072  8E      TEST20  GLO   RE        Get character.
0073  FF 20           SMI   "20"      Test character. Valid?
0075  3B 33           BM    RTN2      No, less than 20 hex.
0077  8E              GLO   RE        Get character.
0078  FF 60           SMI   "60"      Valid character?
007A  33 33           BPZ   RTN2      No, gtr than 5F hex.
007C  9D              GHI   RD        Yes, save RD.
007D  73              STXD            -
007E  8D              GLO   RD        -
007F  73              STXD            -
0080  9C              GHI   RC        Save RC.
0081  73              STXD            -
0082  8C              GLO   RC        -
0083  73              STXD            -
0084  9B              GHI   RB        Save RB.
0085  73              STXD            -
0086  8B              GLO   RB        -
0087  73              STXD            -
0088  8E              GLO   RE        Get character.
0089  52              STR   R2        Push onto stack.
008A  F8 02           LDI   "02"      RC points to character
008C  BC              PHI   RC          pattern table.
008D  F8 FO           LDI   "FO"      -
008F  AC              PLO   RC        -
0090  02              LDN   R2        Get character.
0091  F6              SHR             Character/2.
0092  22              DEC   R2        Save value/2.
0093  52              STR   R2        -
0094  8C              GLO   RC        RC point to character
0095  F4              ADD               pattern.
0096  AC              PLO   RC        -
0097  12              INC   R2        R2 point to saved char.
0098  F8 05           LDI   05        RD = line counter.
009A  AD              PLO   RD        -
009B  02      GETPAT  LDN   R2        Get character.
009C  F6              SHR             Character/2.
009D  33 A6           BDF   RTPAT     If odd, branch.
009F  OC              LDN   RC        Else even, load char pattern.
00A0  F6              SHR             Shift upper nybble
00A1  F6              SHR               into lower nybble.
00A2  F6              SHR             -
00A3  F6              SHR             -
00A4  30 A9           BR    OUTPAT    Branch to output pattern.
00A6  OC      RTPAT   LDN   RC        Get character pattern.
00A7  FA OF           ANI   "OF"      Mask off upper bits.
00A9  22      OUTPAT  DEC   R2        Save char pattern.
00AA  52              STR   R2        -
00AB  22              DEC   R2        -
00AC  52              STR   R2        -
00AD  9E              GHI   RE        Account for space
00AE  FF 01           SMI   "01"        between characters.
00B0  AE              PLO   RE        -
00B1  33 BA           BPZ   TSTCNT    Branch if any shifts left.
00B3  8E      SHIFT8  GLO   RE        Else no shifts left
00B4  FC 08           ADI   "08"      Adjust nbr shifts available.
00B6  AE              PLO   RE        -
00B7  1F              INC   RF        Line addr plus 1.
00B8  C4              NOP
00B9  C4              NOP
00BA  8E      TSTCNT  GLO   RE        Get nbr shifts available.
00BB  32 B3           BZ    SHIFT8    If zero, branch to adjust.
00BD  8E              GLO   RE        Room for a full character?
00BE  FF 03           SMI   "03"      -
00C0  AE              PLO   RE        -
00C1  33 E4           BPZ   SPACE3    Yes, see if exactly 3 spaces.
00C3  02      SHLPAT  LDN   R2        No, less than 3 spaces.
00C4  F6              SHR             Shift pattern left.
00C5  52              STR   R2        Save adjusted pattern.
00C6  1E              INC   RE        Shift counter plus 1.
00C7  8E              GLO   RE        All shifts done?
00C8  3A C3           BNZ   SHLPAT    No, branch and shift again.
00CA  OF              LDN   RF        Yes, store pattern on screen.
00CB  F1              OR              OR pattern to screen byte.
00CC  5F              STR   RF        Store byte.
00CD  1F              INC   RF        Line addr plus 1.
00CE  2E              DEC   RE        Adjust shift counter.
00CF  9E              GHI   RE        -
00D0  FC 04           ADI   "04"      -
00D2  AE              PLO   RE        -
00D3  12              INC   R2        Get original dot pat.
00D4  02              LDN   R2        -
00D5  22              DEC   R2        -
00D6  52              STR   R2        -
00D7  02      SHIRST  LDN   R2        Shift out displayed
00D8  FE              SHL               bits.
00D9  52              STR   R2        -
00DA  2E              DEC   RE        All bits shifted?
00DB  8E              GLO   RE        -
00DC  3A D7           BNZ   SHIRST    No, branch.
```

```
00DE  0F              LDN   RF       Yes, put char on
00DF  F1              OR               screen.
00E0  5F              STR   RF       -
00E1  2F              DEC   RF       Restore for next bits.
00E2  30 F0           BR    NXTDOT
00E4  32 ED   SPACE3  BZ    STRPAT   Branch if no shifts.
00E6  02      SHL2    LDN   R2       Else shift char pat
00E7  FE              SHL              as needed.
00E8  52              STR   R2       -
00E9  2E              DEC   RE       Any shifts left?
00EA  8E              GLO   RE       -
00EB  3A E6           BNZ   SHL2     Yes, branch.
00ED  0F      STRPAT  LDN   RF       No, store char pat
00EE  F1              OR               on screen.
00EF  5F              STR   RF       -
00F0  8F      NXTDOT  GLO   RF       Next dot line down.
00F1  FC 08           ADI   "08"     -
00F3  AF              PLO   RF       -
00F4  8C              GLO   RC       Get next dot pattern
00F5  FC 20           ADI   "20"       in table.
00F7  AC              PLO   RC       -
00F8  12              DEC   R2       Restore stack pointer.
00F9  12              DEC   R2       -
00FA  C0 02 B4        LBR   CONTNU   Branch to cont process.
00FD  C4              NOP
00FE  C4              NOP
00FF  C4              NOP

0100 - 01FF                          Screen display area.
0200 - 02B3                          Character dot table.

02B4  9E      CONTNU  GHI   RE       Get original shifts.
02B5  FF 02           SMI   "02"     Room for full char?
02B7  33 BA           BPZ   TSTLN    Yes, branch.
02B9  2F              DEC   RF       No, restore screen.
02BA  2D      TSTLN   DEC   RD       All char lines on
02BB  8D              GLO   RD         screen?
02BC  CA 00 9B        LBNZ  GETPAT   No, branch.
02BF  8F              GLO   RF       Restore char addr.
02C0  FF 28           SMI   "28"     -
02C2  AF              PLO   RF       -
02C3  9E              GLO   RE       Account for char
02C4  FF 04           SMI   "04"       and space.
02C6  BE              PHI   RE       -
02C7  33 CE           BPZ   SKPADJ   Branch if any shifts
02C9  1F              INC   RF         left, else adjust
02CA  9E              GHI   RE         char addr & shifts.
02CB  FC 08           ADI   "08"     -
02CD  BE              PHI   RE       -
02CE  F8 03   SKPADJ  LDI   "03"     RB points to BIGCH
02D0  BB              PHI   RB         subroutine.
02D1  F8 67           LDI   "67"     -
02D3  AB              PLO   RB       -
02D4  02              LDN   R2       Get character.
02D5  FF 4E           SMI   "4E"     = "N"?
02D7  C2 03 08        LBZ   LAST     Yes, branch.
02DA  02      TESTM   LDN   R2       No, get character.
02DB  FF 4D           SMI   "4D"     = "M"?
02DD  3A E8           BNZ   TEST#    No, branch.
02DF  F8 A0           LDI   "A0"     Yes, RC points to
02E1  AC              PLO   RC         rest of "M".
02E2  79              MARK           Save R(P), R(X).
02E3  DB              SEP   RB       Jmp subroutine BIGCH.
02E4  22              DEC   R2       Restore stack pointer.
02E5  C0 03 08        LBR   LAST
02E8  02      TEST#   LDN   R2       Get character.
02E9  FF 23           SMI   "23"     = "#"?
02EB  3A FC           BNZ   TESTW    No, branch.
02ED  F8 A5           LDI   "A5"     RC points to part
02EF  AC              PLO   RC         of "#" character.
02F0  79              MARK           Save P, X.
02F1  DB              SEP   RB       Jmp subroutine BIGCH.
02F2  22              DEC   R2       Restore stack.
02F3  F8 AF           LDI   "AF"     RC points to rest
02F5  AC              PLO   RC         of "#".
02F6  79              MARK           Save P, X.
02F7  DB              SEP   RB       Jmp subroutine BIGCH.
02F8  22              DEC   R2       Restore stack.
02F9  C0 03 0E        LBR   ENDCH
02FC  02      TESTW   LDN   R2       Get character.
02FD  FF 57           SMI   "57"     = "W"?
02FF  CA 03 0E        LBNZ  ENDCH    No, branch.
0302  F8 AA           LDI   "AA"     RC points to part
0304  AC              PLO   RC         of "W".
0305  79              MARK           Save P, X.
0306  DB              SEP   RB       Jmp subroutine BIGCH.
0307  22              DEC   R2       Restore stack.
0308  F8 A5   LAST    LDI   "A5"     RC points to rest
030A  AC              PLO   RC         of "W".
030B  79              MARK           Save P, X.
030C  DB              SEP   RB       Jmp subroutine BIGCH.
030D  22              DEC   R2       Restore stack.
030E  60      ENDCH   IRX            Inc stack.
030F  72              LDXA           Restore RB.
0310  AB              PLO   RB       -
0311  72              LDXA           -
0312  BB              PHI   RB       -
0313  72              LDXA           Restore RC.
0314  AC              PLO   RC       -
0315  72              LDXA           -
0316  BC              PHI   RC       -
0317  72              LDXA           Restore RD.
0318  AD              PLO   RD       -
0319  F0              LDX            -
031A  BD              PHI   RD       -
```

```
031B  8F                GLO  RF       Get char addr.
031C  FA OF             ANI  "OF"     Mask off upper bits.
031E  52                STR  R2       Save for later test.
031F  FF 07             SMI  "07"     End of a line?
0321  32 28             BZ   EOL      Yes, branch.
0323  02                LDN  R2       No, get character.
0324  FF OF             SMI  "OF"     End of a line?
0326  3A 34             BNZ  TSTEOP   No, branch.
0328  9E        EOL     GHI  RE       Yes, get shifts.
0329  FF 06             SMI  "06"     Room for another char?
032B  33 34             BPZ  TSTEOP   Yes, branch.
032D  8F                GLO  RF       Get char addr.
032E  FC 29             ADI  "29"     Char addr = new line.
0330  AF                PLO  RF       -
0331  F8 09     SHF9    LDI  "09"     Nbr shifts = 9
0333  BE                PHI  RE       -
0334  8F        TSTEOP  GLO  RF       Get char addr.
0335  FF C7             SMI  "C7"     End of page?
0337  CB 00 33          LBM  RTN2     No, branch to return.
033A  22        SCROLL  DEC  R2       Yes, scroll up 1 line.
033B  F8 01             LDI  "01"     RF points to 1st line
033D  BF                PHI  RF         to be scrolled up.
033E  F8 30             LDI  "30"     -
0340  AF                PLO  RF       -
0341  0F        BYTEUP  LDN  RF       Get byte...
0342  52                STR  R2         save it.
0343  8F                GLO  RE       RF points to upper
0344  FF 30             SMI  "30"       line.
0346  AF                PLO  RF       -
0347  02                LDN  R2       Get saved byte.
0348  5F                STR  RF       Store on upper line.
0349  8F                GLO  RF       RF points to lower
034A  FC 30             ADI  "30"       line.
034C  AF                PLO  RF       -
034D  1F                INC  RF       RF points to next byte.
034E  8F                GLO  RF       All lines scrolled?
034F  FF E8             SMI  "E8"     -
0351  3B 41             BM   BYTEUP   No, branch.
0353  F8 C0             LDI  "C0"     Yes, RF points to last
0355  AF                PLO  RF         line on screen.
0356  F8 00     SPACE1  LDI  "00"     Blank out last line...
0358  5F                STR  RF       Store space on screen.
0359  1F                INC  RF       Point to next addr.
035A  8F                GLO  RF       Get screen addr.
035B  FF FE             SMI  "FE"     End of screen?
035D  3B 56             BM   SPACE1   No, branch.
035F  F8 C0             LDI  "C0"     Yes, RF points to
0361  AF                PLO  RF         last line on acreen.
0362  12                INC  R2       Restore stack pointer.
0363  30 31             BR   SHF9     Branch to restore max
                                        nbr of shifts.
0365  12        RTN3    INC  R2       Restore stack pointer.
0366  70                RET           Return to caller.
0367  E2        BIGCH   SEX  R2       R2 = stack pointer.
0368  9E                GHI  RE       Any shifts left?
0369  3A 71             BNZ  SHFT-1   Yes, branch.
036B  1F                INC  RF       No, acreen addr + 1.
036C  F8 07             LDI  "07"     Set nbr of shifts max.
036E  BE                PHI  RE       -
036F  30 75             BR   SETRD
0371  9E        SHFT-1  GHI  RE       Acct for last part of
0372  FF 01             SMI  "01"       big character.
0374  BE                PHI  RE       -
0375  F8 05     SETRD   LDI  "05"     RD is char line cntr.
0377  AD                PLO  RD       -
0378  9E        OUTBIG  GHI  RE       Initialize shft cntr.
0379  AE                PLO  RE       -
037A  22                DEC  R2       Stack - 1.
037B  4C                LDA  RC       Get last part of char.
037C  52                STR  R2       Save.
037D  8E                GLO  RE       Any shifts left?
037E  32 87             BZ   STRBIG   No, branch.
0380  02        SHFBIG  LDN  R2       Shift last part
0381  FE                SHL             next to char.
0382  52                STR  R2       -
0383  2E                DEC  RE       -
0384  8E                GLO  RE       Done?
0385  3A 80             BNZ  SHFBIG   No, branch.
0387  0F        STRBIG  LDN  RF       Yes, store char last
0388  F1                OR              on screen.
0389  5F                STR  RF       -
038A  8F                GLO  RF       RF points to next
038B  FC 08             ADI  "08"       char line.
038D  AF                PLO  RF       -
038E  12                INC  R2       Restore stack pointer.
038F  2D                DEC  RD       All of char on screen?
0390  8D                GLO  RD       -
0391  3A 78             BNZ  OUTBIG   No, branch.
0393  8F                GLO  RF       Else, yes, restore RF.
0394  FF 28             SMI  "28"     -
0396  AF                PLO  RF       -
0397  30 65             BR   RTN3     Branch to return.
0399  F8 06     MAIN    LDI  "06"     R9 points to GETHEX.
039B  B9                PHI  R9       -
039C  F8 46             LDI  "46"     -
039E  A9                PLO  R9       -
039F  F8 06             LDI  "06"     RB points to OUTMSG.
03A1  BB                PHI  RB       -
03A2  F8 53             LDI  "53"     -
03A4  AB                PLO  RB       -
03A5  F8 00             LDI  "00"     RA points to OUTCH.
03A7  BA                PHI  RA       -
03A8  F8 35             LDI  "35"     -
03AA  AA                PLO  RA       -
03AB  E2                SEX  R2       R2 = stack pointer.
```

```
03AC  69                  INP1    Turn on TV chip.
03AD  79                  MARK    Save R(P), R(X).
03AE  DB                  SEP   RB    Jmp subroutine OUTMSG.
03AF  0D 4C 4F   MSG1             Message 1:
03B2  41 44 20                      "LOAD START ADDR".
03B5  53 54 41                    -
03B8  52 54 20                    -
03BB  41 44 44                    -
03BE  52 00                       00 = end of message.
03C0  22                  DEC   R2    Control returns here,
                                        restore stack pointer.
03C1  D9                  SEP   R9    Jmp subroutine GETHEX.
                                        Input a byte.
03C2  BD                  PHI   RD    RD points to start addr.
03C3  D9                  SEP   R9    -
03C4  AD                  PLO   RD    -
03C5  79                  MARK        Save R(P), R(X).
03C6  DB                  SEP   RB    Jmp subroutine OUTMSG.
03C7  4C 4F 41   MSG2             Message 2:
03CA  44 20 45                      "LOAD END ADDR".
03CD  4E 44 20                    -
03D0  41 44 44                    -
03D3  52 00                       -
03D5  22                  DEC   R2    Restore R(X).
03D6  D9                  SEP   R9    Jmp subroutine GETHEX.
03D7  BC                  PHI   RC    RC points to end addr.
03D8  D9                  SEP   R9    -
03D9  AC                  PLO   RC    -
03DA  79                  MARK        Save R(P), R(X).
03DB  DB                  SEP   DB    Jmp subroutine OUTMSG.
03DC  0D 41 44   MSG3             Message 3:
03DF  44 52 20                      "ADDR INST  OP".
03E2  49 4E 53                    -
03E5  54 20 20                    -
03E8  4F 50 00                    -
03EB  22                  DEC   R2    Restore stack pointer.
03EC  3F EC      KEYPR    BN4   KEYPR   Loop till key pressed.
03EE  37 EE      KEYREL   B4    KEYREL  Loop till released.
03F0  8C                  GLO   RC    Get end addr.
03F1  52                  STR   R2    Save.
03F2  8D                  GLO   RD    Get curr code addr.
03F3  F7                  SM          End addr - curr addr.
03F4  3B FA               BM    NOTEND  If not end, branch.
03F6  9C                  GHI   RC    Test addr hi for end.
03F7  52                  STR   R2    -
03F8  9D                  GHI   RD    -
03F9  F7                  SM          -
03FA  CB 04 0C   NOTEND   LBM   NUCODE  Not end, output another
                                          line of code.
03FD  79         END      MARK        Save R(P), R(X).
03FE  DB                  SEP   RB    Jmp subroutine OUTMSG.
03FF  0A 45 4E   MSG4             Message 4: "END".
0402  44 00                       -
0404  22                  DEC   R2    Restore R(X).
0405  3F 05      KEYPR2   BN4   KEYPR2  Loop till key pressed.
0407  37 07      KEYRL2   B4    KEYRL2  Loop till released.
0409  C0 03 99            LBR   MAIN    Branch, start over.
040C  F8 0A               LDI   "0A"    Output Line Feed.
040E  79                  MARK        Save X, P.
040F  DA                  SEP   RA    Jmp subroutine OUTCH.
0410  22                  DEC   R2    Restore stack.
0411  F8 06               LDI   "06"    R6 points to OUTDGT.
0413  B8                  PHI   R8    -
0414  F8 5D               LDI   "5D"    -
0416  A8                  PLO   R8    -
0417  9D                  GHI   RD    Get curr addr hi.
0418  F6                  SHR         Left nybble into
0419  F6                  SHR           right nybble.
041A  F6                  SHR         -
041B  F6                  SHR         -
041C  D8                  SEP   R8    Jmp subroutine OUTDGT.
041D  9D                  GHI   RD    Get curr addr hi.
041E  FA 0F               ANI   "0F"    Mask off upper bits.
0420  D8                  SEP   R8    Jmp subroutine OUTDGT.
0421  8D                  GLO   RD    Get curr addr low.
0422  F6                  SHR         Left nybble into right.
0423  F6                  SHR         -
0424  F6                  SHR         -
0425  F6                  SHR         -
0426  D8                  SEP   R8    Jmp subroutine OUTDGT.
0427  8D                  GLO   RD    Get curr addr low.
0428  FA 0F               ANI   "0F"    Mask off upper bits.
042A  D8                  SEP   R8    Jmp subroutine OUTDGT.
042B  F8 20               LDI   "20"    Output a space.
042D  79                  MARK        Save P, X.
042E  DA                  SEP   RA    Jmp subroutine OUTCH.
042F  22                  DEC   R2    Restore stack.
0430  0D                  LDN   RD    Get op code.
0431  3A 40               BNZ   TESTON  If not 00, branch.
0433  79                  MARK        Else save X, P.
0434  DB                  SEP   RB    Jmp subroutine OUTMSG.
0435  49 44 4C   MSG5             Message 5: "IDL".
0438  20 00
043A  22                  DEC   R2    Restore stack.
043B  C4                  NOP
043C  1D         NEXTOP   INC   RD    Point to next op code.
043D  C0 03 EC            LBR   KEYPR   Branch to output new line.
0440  0D         TESTON   LDN   RD    Get op code.
0441  F6                  SHR         Get instruction bits.
0442  F6                  SHR         -
0443  F6                  SHR         -
0444  F6                  SHR         -
0445  52                  STR   R2    Save.
0446  3A 5E               BNZ   TEST1N  Branch if not ON op.
0448  79                  MARK        Save P, X.
```

```
0449  DB                   SEP   RB       Jmp subroutine OUTMSG.
044A  4C 44 4E   MSG6                     Message 6: "LDN".
044D  20 20 00                            -
0450  22         OUTNBR    DEC   R2       Restore stack.
0451  C4                   NOP
0452  C4                   NOP
0453  F8 52                LDI   "52"     Output "R".
0455  79                   MARK
0456  DA                   SEP   RA       Jmp subroutine OUTCH.
0457  22         OUTREG    DEC   R2       Restore stack.
0458  0D                   LDN   RD       Get op code.
0459  FA 0F                ANI   "0F"     Mask off upper bits.
045B  D8                   SEP   R8       Output reg nbr.
045C  30 3C                BR    NEXTOP   Branch to INC RD.
045E  02         TEST1N    LDN   R2       Get op code.
045F  FF 01                SMI   "01"     Op code = 1?
0461  3A 72                BNZ   TEST2N   No, branch.
0463  79                   MARK           Else, yes.
0464  DB                   SEP   RB       Out msg 7.
0465  49 4E 43   MSG7                     Message 7: "INC".
0468  20 20 00                            -
046B  22                   DEC   R2       Restore stack.
046C  F8 52                LDI   "52"     Output "R".
046E  79                   MARK           -
046F  DA                   SEP   RA       -
0470  30 57                BR    OUTREG   Br to out reg nbr.
0472  02         TEST2N    LDN   R2       Get op.
0473  FF 02                SMI   "02"     Op = 2N?
0475  3A 81                BNZ   TEST4N   No, branch.
0477  79                   MARK           Yes.
0478  DB                   SEP   RB       Jmp SR OUTMSG.
0479  44 45 43   MSG8                     Message 8: "DEC".
047C  20 20 00                            -
047F  30 50                BR    OUTNBR   Br to out reg nbr.
0481  02         TEST4N    LDN   R2       Op code = 4N?
0482  FF 04                SMI   "04"     -
0484  3A 90                BNZ   TEST5N   No, branch.
0486  79                   MARK           Yes.
0487  DB                   SEP   RB       Out "LDA".
0488  4C 44 41   MSG8                     -
048B  20 20 00                            -
048E  30 50                BR    OUTNBR   Out reg nbr.
0490  02         TEST5N    LDN   R2       Get op.
0491  FF 05                SMI   "05"     Op = 5N?
0493  3A 9F                BNZ   TEST6N   No, branch.
0495  79                   MARK           Yes, out "STR".
0496  DB                   SEP   RB       -
0497  53 54 52   MSG9                     -
049A  20 20 00                            -
049D  30 50                BR    OUTNBR   -
049F  02                   LDN   R2       Get op.
04A0  FF 06                SMI   "06"     Op code = 6N?
04A2  3A C8                BNZ   TEST8N   No, branch.
04A4  0D                   LDN   RD       Yes, "INP" or "OUT"?
04A5  FA 0F                ANI   "0F"     Mask upper bits.
04A7  52                   STR   R2       Save.
04A8  FF 08                SMI   "08"     "INP"?
04AA  33 B9                BPZ   NOTINP   No, branch.
04AC  79                   MARK           Else, "OUT".
04AD  DB                   SEP   RB       Jmp SR OUTMSG.
04AE  4F 55 54   MSG10                    -
04B1  20 20 00                            -
04B4  C4                   NOP
04B5  C4                   NOP
04B6  C4                   NOP
04B7  30 C1                BR    OUTPRT   Br to out port nbr.
04B9  79         NOTINP    MARK
04BA  DB                   SEP   RB       Jmp SR OUTMSG.
04BB  49 4E 50   MSG11                    Out "INP".
04BE  20 20 00                            -
04C1  22         OUTPRT    DEC   R2       Restore stack.
04C2  0D                   LDN   RD       Get op.
04C3  FA 07                ANI   "07"     Mask upper bits.
04C5  D8                   SEP   R8       Jmp SR OUTDGT.
04C6  30 3C                BR    NEXTOP   Get next op.
04C8  02         TEST8N    LDN   R2       Get op.
04C9  FF 08                SMI   "08"     Op = 8N?
04CB  3A E0                BNZ   TEST9N   No, branch.
04CD  79                   MARK           Yes, out "GLO".
04CE  DB                   SEP   RB       -
04CF  47 4C 4F   MSG12                    -
04D2  20 20 00                            -
04D5  22                   DEC   R2       Restore stack.
04D6  C4                   NOP
04D7  C4                   NOP
04D8  C4                   NOP
04D9  F8 52                LDI   "52"     Out "R".
04DB  79                   MARK           -
04DC  DA                   SEP   RA       Jmp SR OUTCH.
04DD  C4                   NOP            -
04DE  30 57                BR    OUTREG   -
04E0  02         TEST9N    LDN   R2       Get op.
04E1  FF 09                SMI   "09"     Op = 9N?
04E3  3A EF                BNZ   TESTAN   No, branch.
04E5  79                   MARK           Yes, out "GHI".
04E6  DB                   SEP   RB       -
04E7  47 48 49   MSG13                    -
04EA  20 20 00                            -
04ED  30 50                BR    OUTNBR   -
04EF  02         TESTAN    LDN   R2       Get op.
04F0  FF 0A                SMI   "0A"     Op = AN?
04F2  3A FE                BNZ   TESTBN   No, branch.
04F4  79                   MARK           Yes, out "PLO".
04F5  DB                   SEP   RB       -
04F6  50 4C 4F   MSG14                    -
```

```
04F9  20 20 00                               -
04FC  30 50               BR    OUTNBR       -
04FE  02                  LDN   R2           Get op.
04FF  FF 0B               SMI   "0B"         Op = BN?
0501  3A 0E               BNZ   TESTDN       No, branch.
0503  79                  MARK               Yes, out "PHI".
0504  DB                  SEP   RB           -
0505  50 48 49   MSG15                       -
0508  20 20 00                               -
050B  C0 04 50            LBR   OUTNBR       -
050E  02         TESTDN   LDN   R2           Get op.
050F  FF 0D               SMI   "0D"         Op = DN?
0511  3A 25               BNZ   TESTEN       No, branch.
0513  79                  MARK               Yes, out "SEP".
0514  DB                  SEP   RB           Jmp SR OUTMSG.
0515  53 45 50   MSG16                       -
0518  20 20 00                               -
051B  22         OUTEQL   DEC   R2           -
051C  79                  MARK               Output "=R".
051D  DB                  SEP   RB           -
051E  3D 52 00   MSG17                       -
0521  C4                  NOP                -
0522  C0 04 57            LBR   OUTREG       Out reg nbr.
0525  02         TESTEN   LDN   R2           Get op.
0526  FF 0E               SMI   "0E"         Op = EN?
0528  3A 34               BNZ   TEST3N       No, branch.
052A  79                  MARK               Yes, out "SEX".
052B  DB                  SEP   RB           -
052C  53 45 58   MSG18                       -
052F  20 20 00                               -
0532  30 1B               BR    OUTEQL       Br to output "=R".
0534  C4         TEST3N   NOP
0535  C4                  NOP
0536  C4                  NOP
0537  02                  LDN   R2           Get op.
0538  FF 03               SMI   "03"         Op = 3N?
053A  3A 82               BNZ   TEST7N       No, branch.
053C  0D                  LDN   RD           Get op code.
053D  FF 38               SMI   "38"         Op = "SKP"?
053F  3A 4D               BNZ   OUTBR        No, branch.
0541  79                  MARK               Yes, out "SKP".
0542  DB                  SEP   RB           -
0543  53 4B 50   MSG19                       -
0546  00                                     -
0547  C4                  NOP                -
0548  C4                  NOP                -
0549  22                  DEC   R2           -
054A  C0 04 3C   LBRNXT   LBR   NEXTOP       Br to get more code.
054D  F8 42      OUTBR    LDI   "42"         Out "B".
054F  79                  MARK               -
0550  DA                  SEP   RA           -
0551  22                  DEC                -
0552  F8 07               LDI   "07"         R7 points to 3N table.
0554  B7                  PHI   R7           -
0555  0D                  LDN   RD           Get op.
0556  FA 0F               ANI   "0F"         Mask off upper bits
0558  A7                  PLO   R7             to point to mnemonic.
0559  07                  LDN   R7           Get character.
055A  79                  MARK               Out character.
055B  DA                  SEP   RA           -
055C  22                  DEC   R2           -
055D  87                  GLO   R7           R7 point to next char.
055E  FC 10               ADI   "10"         -
0560  A7                  PLO   R7           -
0561  07                  LDN   R7           Get next char.
0562  79                  MARK               Out next char.
0563  DA                  SEP   RA           -
0564  22                  DEC   R2           -
0565  79                  MARK               Out 2 spaces.
0566  DB                  SEP   RB           -
0567  20 20 00   MSG20                       -
056A  22                  DEC   R2           -
056B  9D                  GHI   RD           Get curr addr.
056C  F6                  SHR                Out hi order digit.
056D  F6                  SHR                -
056E  F6                  SHR                -
056F  F6                  SHR                -
0570  D8                  SEP   D8           Jmp SR OUTDGT.
0571  9D                  GHI   RD           Out 2nd addr digit.
0572  FA 0F               ANI   "0F"         -
0574  D8                  SEP   R8           Jmp SR OUTDGT.
0575  1D         OUTADD   INC   RD           Out branch addr.
0576  0D                  LDN   RD           Out br addr hi digit.
0577  F6                  SHR                -
0578  F6                  SHR                -
0579  F6                  SHR                -
057A  F6                  SHR                -
057B  D8                  SEP   R8           -
057C  0D                  LDN   RD           Out br addr low digit.
057D  FA 0F               ANI   "0F"         -
057F  D8                  SEP   R8           -
0580  30 4A               BR    LBRNXT       Br to get more code.
0582  02         TEST7N   LDN   R2           Get op.
0583  FF 07               SMI   "07"         Op = 7N?
0585  3A B0               BNZ   TESTCN       No, branch.
0587  F8 07               LDI   "07"         Yes, out 7N mnemonic.
0589  B7                  PHI   R7           R7 points to 7N table.
058A  0D                  LDN   RD           Get op.
058B  FA 0F               ANI   "0F"         Mask upper bits.
058D  FC 20               ADI   "20"         Adjust R7 low.
058F  A7                  PLO   R7           -
0590  F8 04               LDI   "04"         R6 = char counter.
0592  A6                  PLO   R6           -
0593  07         OUT7N    LDN   R7           Get char.
0594  79                  MARK               Out char.
```

```
0595  DA                 SEP   RA       -
0596  22                 DEC   R2       -
0597  87                 GLO   R7       R7 points to next
0598  FC 10              ADI   "10"       character.
059A  A7                 PLO   R7       -
059B  26                 DEC   R6       Char cntr minus 1.
059C  86                 GLO   R6       Test cntr. Done?
059D  3A 93              BNZ   OUT7N    No, branch.
059F  F8 20              LDI   "20"     Yes, out a space.
05A1  79                 MARK           -
05A2  DA                 SEP   RA       -
05A3  22                 DEC   R2       -
05A4  0D                 LDN   RD       Get op.
05A5  FF 7E              SMI   "7E"     Op = "RSHL"?
05A7  32 4A              BZ    LBRNXT   Yes, branch.
05A9  0D                 LDN   RD       No, test.
05AA  FF 7C              SMI   "7C"     Op less than "ADCI"?
05AC  3B 4A              BM    LBRNXT   Yes, branch.
05AE  30 75              BR    OUTADD   No, br to out addr.
05B0  02        TESTCN   LDN   R2       Get op.
05B1  FF 0C              SMI   "0C"     Op = CN?
05B3  CA 06 18           LBNZ  TESTFN   No, branch.
05B6  C4                 NOP            Else yes, out mnemonic.
05B7  C4                 NOP            -
05B8  0D                 LDN   RD       Get op.
05B9  FF C4              SMI   "C4"     Op = "NOP"?
05BB  3A C6              BNZ   OUTCN    No, branch.
05BD  79                 MARK           Yes, out "NOP".
05BE  DB                 SEP   RB       -
05BF  4E 4F 50  MSG21                   -
05C2  00                                -
05C3  22                 DEC   R2       -
05C4  30 4A              BR    LBRNXT   Br to get next op.
05C6  F8 4C     OUTCN    LDI   "4C"     Out "L".
05C8  79                 MARK           -
05C9  DA                 SEP   RA       -
05CA  22                 DEC   R2       -
05CB  F8 07              LDI   "07"     R7 points to CN table.
05CD  B7                 PHI   R7       -
05CE  0D                 LDN   RD       -
05CF  FA 0F              ANI   "0F"     -
05D1  FC 60              ADI   "60"     -
05D3  A7                 PLO   R7       -
05D4  F8 03              LDI   "03"     R6 = character cntr.
05D6  A6                 PLO   R6       -
05D7  07        OUTCOP   LDN   R7       Get character.
05D8  79                 MARK           Out char.
05D9  DA                 SEP   RA       -
05DA  22                 DEC   R2       -
05DB  87                 GLO   R7       R7 point to next char.
05DC  FC 10              ADI   "10"     -
05DE  A7                 PLO   R7       -
05DF  26                 DEC   R6       Char cntr - 1.
05E0  86                 GLO   R6       All chars out?
05E1  3A D7              BNZ   OUTCOP   No, branch.
05E3  F8 20              LDI   "20"     Cut a space.
05E5  79                 MARK           -
05E6  DA                 SEP   RA       -
05E7  22                 DEC   R2       -
05E8  C4                 NOP            -
05E9  0D                 LDN   RD       Get op.
05EA  FF C5              SMI   "C5"     Op = "LSNQ"?
05EC  32 4A              BZ    LBRNXT   Yes, branch.
05EE  0D                 LDN   RD       Get op.
05EF  FF C6              SMI   "C6"     Op = "LSNZ"?
05F1  32 4A              BZ    LBRNXT   Yes, branch.
05F3  0D                 LDN   RD       Get op.
05F4  FF C7              SMI   "C7"     Op = "LSNF"?
05F6  32 4A              BZ    LBRNXT   Yes, branch.
05F8  0D                 LDN   RD       Get op.
05F9  FF C8              SMI   "C8"     Op = "LSKP"?
05FB  32 4A              BZ    LBRNXT   Yes, branch.
05FD  0D                 LDN   RD       Get op.
05FE  FF CC              SMI   "CC"     Op less than "LSIE"?
0600  C3 05 4A           LBPZ  LBRNXT   No, branch. Done.
0603  F8 02              LDI   "02"     Yes, R6 = byte cntr.
0605  A6                 PLO   R6       -
0606  1D        OUTAD2   INC   RD       Get long branch addr hi.
0607  0D                 LDN   RD       -
0608  F6                 SHR            Out hi order digit.
0609  F6                 SHR            -
060A  F6                 SHR            -
060B  F6                 SHR            -
060C  D8                 SEP   R8       -
060D  0D                 LDN   RD       Out low order digit.
060E  FA 0F              ANI   "0F"     -
0610  D8                 SEP   R8       -
0611  26                 DEC   R6       Byte cntr - 1.
0612  86                 GLO   R6       Done?
0613  3A 06              BNZ   OUTAD2   No, branch.
0615  C0 05 4A           LBR   LBRNXT   Yes, br to get more code.
0618  F8 07     TESTFN   LDI   "07"     R7 points to FN table.
061A  B7                 PHI   R7       -
061B  0D                 LDN   RD       Get op.
061C  FA 0F              ANI   "0F"     Adjust pointer byte low.
061E  FC 90              ADI   "90"     -
0620  A7                 PLO   R7       -
0621  F8 03              LDI   "03"     R6 = character counter.
0623  A6                 PLO   R6       -
0624  07        OUTFOP   LDN   R7       Get char.
0625  79                 MARK           Out FN mnemonic.
0626  DA                 SEP   RA       -
0627  22                 DEC   R2       -
0628  87                 GLO   R7       Adjust R7. Point
0629  FC 10              ADI   "10"       to next character.
```

```
062B  A7              PLO   R7       -
062C  26              DEC   R6       Char cntr - 1.
062D  86              GLO   R6       Test cntr. Done?
062E  3A 24           BNZ   OUTFOP   No, branch back.
0630  79              MARK           Yes, done. Out 2 spaces.
0631  DB              SEP   RB       -
0632  20 20 00  MSG22                -
0635  22              DEC   R2       -
0636  0D              LDN   RD       Get op.
0637  FF F8           SMI   "F8"     Op less than "LDI"?
0639  CB 04 3C        LBM   NEXTOP   Yes, branch.
063C  0D              LDN   RD       No, get op.
063D  FF FE           SMI   "FE"     Op = "SHL"?
063F  C2 04 3C        LBZ   NEXTOP   Yes, branch.
0642  C0 05 75        LBR   OUTADD   No, branch to out operand.

                   ----SUBROUTINES----

0645  D3        RTN4    SEP   R3       Return to caller.
0646  6C        GETHEX  INP4           Input hex byte.
0647  52                STR   R2       Save.
0648  64                OUT4           Output hex byte to display.
0649  22                DEC   R2       Restore stack.
064A  3F 46             BN4   GETHEX   Loop till INPUT key pressed.
064C  37 4C     KEYRL2  B4    KEYRL2   Loop till INPUT released.
064E  30 45             BR    RTN4     Br to return.

0650  E2        RTN5    SEX   R2       Restore stack pointer.
0651  12                INC   R2       Restore stack.
0652  70                RET            Return to caller.
0653  F0        OUTMSG  LDX            Get character via old R(P).
0654  32 50             BZ    RTN5     If "00", end.  Branch.
0656  72                LDXA           Else, get char and INC R(X).
0657  79                MARK           Out character.
0658  DA                SEP   RA       Jmp SR OUTCH.
0659  22                DEC   R2       Restore stack.
065A  30 53             BR    OUTMSG   Br to get another char.

065C  D3        RTN6    SEP   R3       Return to caller.
065D  FF 0A     OUTDGT  SMI   "0A"     Digit less than 0A hex?
065F  C7                LSM            Yes, skip next instr.
0660  FC 07             ADI   "07"     No, hex to decimal.
0662  FC 3A             ADI   "3A"     Decimal digit to ASCII.
0664  79                MARK           Out character.
0665  DA                SEP   RA       -
0666  22                DEC   R2       -
0667  30 5C             BR    RTN6     Br to return.

0668 - 0700     STACK
```

A/D Converter Design For the TRS-80 Level II

Read voltages from 0 to + 10 volts, and display the results on the video monitor.

Arthur Mullin, Jr.
Rte. 3, Box C-9
Beaumont, TX 77706

This circuit and accompanying software will allow the user to read voltage from 0 to +10 volts and display the results on the video monitor. I am using a 9 volt battery to test the system, but any source of voltage will do as long as the maximum ratings are not exceeded. This opens up a whole world of possibilities, such as temperature monitors or wind-speed monitors.

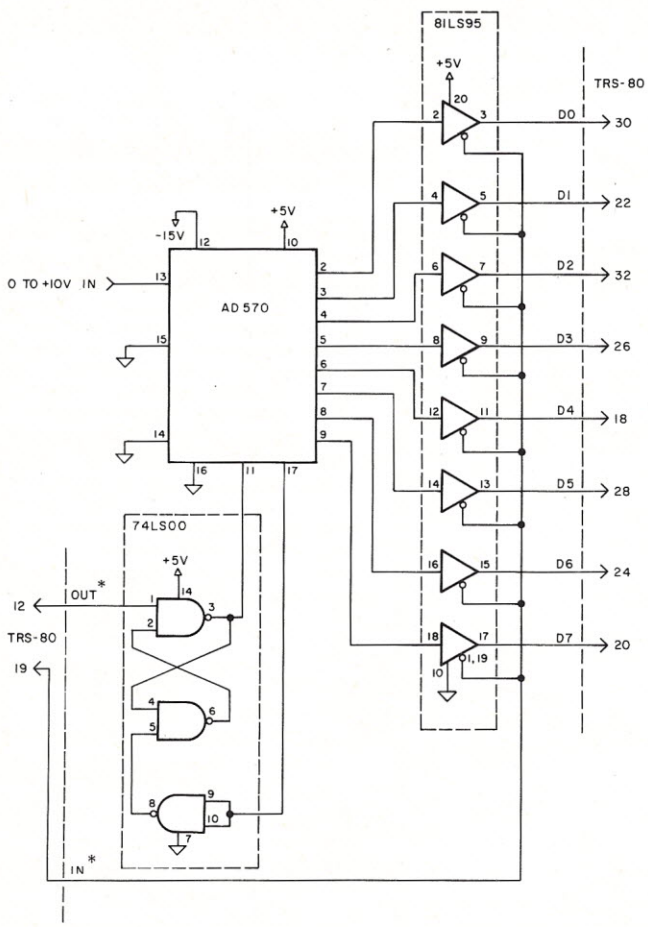


Fig. 1. Assembly schematic.

Fig. 1 shows that this circuit is simple to build since there are only three chips. The heart of the circuit is the analog-to-digital converter AD 570, which can be purchased from Analog Devices (Rte. 1, Industrial Park, PO Box 280, Norwood, MA 02062) for about $23. A quad NAND gate, the 74LS00, and the octal Tri-state non-inverting buffer, 81LS95, can be purchased from a local parts store at nominal prices.

The AD 570

The AD 570 converts a voltage in the range of 0 to + 10 volts into a corresponding binary number and presents this number to the computer. If the input voltage is 0, then the number output is 0. If the input voltage is + 10, then the binary number output is 255. Accordingly, if the voltage is + 5, then the output is 127.

By instructing the computer to divide the binary number it receives by 255, a percentage of full-scale voltage will be obtained. For instance, if the input voltage is + 5 volts, the number the computer receives will be 127. If 127 is divided by 255, the result will be .49803922 (let's call it .5). By multiplying this value by 10, we can read voltage directly on the video monitor. Notice that if we advance the voltage to the converter ever so slightly so as to make the output binary number equal to 128, the result of dividing by 255 will yield an answer of .50196078.

From this, you can see that there are "gaps" in the conversion from analog to digital. This is an inherent characteristic of digitization, and further understanding of this phenomenon can be achieved by reading up on analog-to-digital conversion.

Conversion Cycle

Refer to Fig. 2. The normal state of affairs is, of course, for OUT* to be high (close to 5 volts). NAND gate A will have a low (close to 0 volts) on pin 3. Since pin 3 is connected to pin 4 of gate B, pin 6 is forced high, and pin 2 of gate A is high, further convincing pin 3 of gate A to be low.

At this time, DATA READY* is low, and data is available at pins 2–9 of the A/D converter. If we did not use a Tri-state buffer network to accept this data, the computer would be flaking out about now. Since DATA READY* is low, pin 8 of gate C is high. This further reinforces a high at pin 6 of gate B, since pin 4 is already low.

Since DATA READY* is low and data is available at pins 2–9, there is nothing preventing us from reading data into the computer at this time. All we have to do is force pins 1 and 19 of the 81LS95 to go low. This can be done by typing "PRINT INP(1)" (or any address). The

computer will print whatever is available at pins 2-9 of the converter.

But let's not read the data at the buffers yet. Instead, let's get a new value from the converter and then read the data. To do this, it will be necessary to bring the CONVERT* pin of the AD 570 from its low state to a high state and then back to a low state. I find it more understandable to view this process as giving the converter a "kick."

Let's type "OUT 0,0". Again, any address and any number output (0 to 255) will do, since we're only interested in forcing the OUT* line to go low. When OUT* goes low, pin 3 of NAND gate A goes high, making pin 11 of the converter go high. After about 1.5 microseconds, DATA READY* goes high and no data is available at pins 2-9 of the converter.

When DATA READY* goes high, pin 8 of NAND gate C goes low, forcing pin 6 of gate B to go high. Now pin 2 of gate A is high and OUT* has gone back to its normal high state. This forces pin 3 of gate A back low, taking pin 11 of the converter with it.

Now the converter is going through its conversion cycle, which takes about 25 microseconds. While the converter is busy converting the analog voltage at pin 13 of the converter, DATA READY* is still high and will not go low until the conversion is complete.

If the TRS-80 were an extremely fast machine, we would have to delay any requests for data from the converter at this time, since data is available at pins 2-9 of the converter only when DATA READY* is low. Fortunately, from the standpoint of chip count, we do not have to hold the computer up, and DATA READY* goes low when the converter has completed its task.

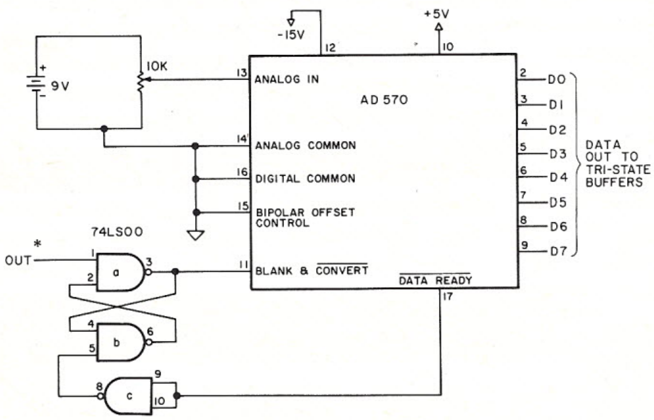


Fig. 2. A/D converter.

Now we are in the state initially described, and the new data is available at pins 2-9. We can ask for this data by requesting an input and enabling the Tristate buffers.

The AD 570 can process +5 and −5 volts by lifting pin 15 from ground. In this mode, −5 volts gives an output of 0, +5 volts gives a binary 255, and 0 volts gives 127 output.

Listing 1 will print the binary output of the converter at the top left side of the video monitor, while displaying the voltage value in the center of the screen.

For those of you wishing more detailed information on conversion techniques in general, or the AD 570 specifically, I suggest you read "Analog-Digital Conversion Notes" or the data sheet on the AD 570. Both can be obtained from Analog Devices. ■

```
10 CLS
20 OUT 1, 1
30 PRINT @0, INP(1)
40 A = (INP(1)/255)*10
50 PRINT @540, A
60 GOTO 20
```

Listing 1.

MICRO-SCOPE

Edited by Dennis Brisson

Hardware and Underwear

A New Haven, Connecticut, manufacturer of electronic instruments and hardware has recently changed its name to Global Specialties Corporation. The name change reflects the changing nature of the company, as company President Ronald J. Portugal explains: "Today, our sales network is truly global in nature. We have expanded our horizons, and in doing so have worked to expand the horizons of the industry. In everything we have set out to do, our goal has been to think broadly, to appeal to the many instead of the few, to offer customers new kinds of tools to make their work easier, less aggravating, less time-consuming. And we have worked to increase the world's understanding of these tools, to show them new ways to use them, to teach them easier ways to get their work done."

Such lofty goals are admirable, but it was such broadmindedness that raised more than a few eyebrows at the company's headquarters in Great Britain. The company's original name was Continental Specialties, which had a more suggestive meaning in England and parts of Europe. It alluded to such esoteric novelties as peek-a-boo lingerie, "French Ticklers" and marital aids.

The "naughty-naughty" implications in the European markets became a growing embarrassment. This may, perhaps, have been compounded by the fact that the company's European operations were conducted by an office staffed almost entirely by women, and headed by a very proper English lady, Mrs. Tina Knight, managing director.

With the name change, Global Specialties continues its overseas operations, but without prompting an embarrassed blush.

Is There a Doctor in the Studio?

One way in which the health care industry is seeking to keep costs down is through the use of videotape as a communications tool to train both patients and staff.

At Tucson General Hospital in Arizona, video equipment helps educate patients, who are as likely to see a program on "Living with Respiratory Illness" as the afternoon movie when they switch on the bedside television. Through video equipment, Tucson General makes medical programs available all day to interested patients. A patient can watch a tape of an operation he will undergo or choose a music program for relaxation before entering surgery.

At the University of Pittsburgh's Western Psychiatric Institute and Clinic (WPIC), educational programs are produced to inform patients about the medicines they are given, test and treatment procedures and available hospital facilities. According to one clinic spokesman, "By taping this information we hope to produce a patient education program that is comprehensive and consistent. Each patient will get the same complete explanation."

In many hospitals, groups of medical students crowd around a video cassette player and monitor, just as they once gathered to peer down into operating rooms from glass-walled balconies. One nursing home director finds video equipment invaluable for training personnel and staff education. Taping lectures eliminates the need to have the education director repeat them constantly and allows reaching evening and night shifts with vital information and lectures that are done live only during the day.

Several hospitals maintain their own audiovisual departments, which are involved in corporate communications, staff education and videotaping operations, diagnostic interviews, etc.

Since nonprofit institutions such as hospitals operate on limited budgets, they have to use their funds as efficiently as possible. More and more hospitals are producing their own films with videotape as a low-cost medical training tool.

Computer Beats Rush-Hour Blues

William Kelly needed a change of pace. He was bored with his daily commute to work and felt that his time could be more productively spent. Kelly, founder and president of SAI Electronics of Farmingdale, Long Island, wished the ability to process data during his commutation time, so he had a complete data-processing center installed in his car.

The unit, installed by Berliner Computer Center of New Hyde Park, NY, includes a 48K RAM Apple computer with 232K of disk storage and three languages resident in ROM. It also features 16-color capability and complete sound synthesis. In addition to the computer unit, the company also installed a stereo system, radar detection units, burglar alarm systems, CB radio with automatic electric antenna, remote control starting system, videotape recorder and player, color video camera and 120 volt ac inverter to allow the use of small appliances, such as electric toothbrush, shaver, etc.

This equipment nicely complements the car's original accessories—sink with running water, soda dispenser, icebox, color TV and bar—which came equipped on Kelly's custom-built 1980 Cadillac Limousine.

Popcorn and Teleporn

Home information systems users may be getting more than they bargained for from the programming services, which normally provide homeowners with information, games and entertainment via specially adapted TV sets or home computers, but may now be spicing up their offerings with electronically transmitted pornographic material. This possibility has prompted speculation that censorship—either voluntary or government imposed—may soon emerge as an issue.

According to *Electronic Mail and Message Systems* (EMMS), recently, viewers of the British Prestel viewdata service were presented with the unexpected possibility of accessing a "dirty books guide." The publicity that surrounded this incident apparently embarrassed Sir William Barlow, chairman of the British Post Office, which imple-

mented this system for home computer usage in 1976 (see *Microcomputing*, October 1979, p. 94). He defended a hands-off policy on viewdata information, claiming that it was analogous to the (private) contents of telephone conversations or mail.

In the United States, other forms of new electronic gadgetry for the home are apparently benefiting, in part, from electronic porn. With about one million videotape recorders now installed in U.S. homes, the market for pornographic videotapes is "very strong," according to the EMMS article, which points to widespread advertising of taped porn in such "skin" magazines as *Playboy*, *Penthouse* and *Forum*.

Cable TV is another rapidly developing conduit for racy material. Subscribers to many CATV services who pay a nominal fee are offered a wide variety of entertainment "specials," including a selection of newly released films that typically include more than one uncut R-rated movie. While most Americans believe that pornography is OK as long as it is in the confines of the home and is out of sight, some executives in the consumer electronics and electronic message/home information industry worry that some of these services will become associated in the public mind with the world of pornography.

MicroNet, a consumer timesharing service, has gone to some lengths to protect their service from porno problems. The CB Simulator channels on MicroNet, which can be used for home computerists to talk to each other, are patrolled regularly by MicroNet staff members. Furthermore, the games available on MicroNet have a "Porno-Trap," which detects the use of obscenities by the game player. Once the computer detects bawdy language, it immediately cautions the player against using such words, and the player is likely to be further reprimanded when he finds a message addressed to him on the electronic "bulletin board" mentioning the incident.

Leaders in the electronic communications field believe that similar types of precautions will be developed and will prevent the electronic message field from becoming swamped in porno.

Today's Army Is Quicker, Lighter, More Efficient & Less Expensive

Computerized micropublishing is now a way of life in the U.S. Army. The Army has converted many of its publications into microfiche, which has significantly cut down on the costs and time required to publish Army indexes. The computer-output micrographics (COM) system has reduced the time necessary to about 30 days—from initial selection of data to mailing—in contrast to the former manual system, which could take as long as 9 months.

Because the Army's indexes now can be updated quickly and accurately, the most immediate effect will be felt worldwide by the more than 50,000 users who rely on the timeliness of the information contained in these publications—indexes to the Army's technical publications, administrative procedures, training and doctrinal publications, obsolete publications and an index of blank forms.

In addition to the cost savings and time factors, the microfiche system reduces the warehousing space required because the process uses no paper and eliminates the need for bulk handling and storage. An index of 1000 pages, for example, weighing several pounds in paper, weighs less than one ounce in microfiche form. Consequently, mailing rates, too, are drastically reduced. And, because of the swiftness, accuracy and efficiency of the microfiche method, the five indexes can now be published on a quarterly schedule.

With the new system, it is estimated that savings to the Army will be more than $5 million over the previous system of publishing hard copy.

Watch for Tiny Bulges

Employees at IC manufacturing companies in Silicon Valley, California, may leave work with a chip on their shoulders, but if they leave with chips in their pockets, socks or shirt sleeves, they may get caught. According to *Newsweek* magazine (May 12, 1980), most IC companies are tightening security measures in an attempt to stem the incidence of chip loss due to employee stealing. One IC, the article states, can sell on the black market for as much as $100.

Such a black market exists and is thriving because chip manufacturers are unable to keep up with the demand for these silicon wafers. *Newsweek* states that industry officials hope to balance supply and demand by increasing production capabilities.

Computer-Aided Aim

Hunters and fishermen are now better prepared to take to the wild to bag their favorite trophy after viewing the "Fish and Game Forecaster," a TV news insert that predicts when hunting and fishing conditions will be at their peak. The forecast is based on a computerized system that monitors and predicts optimum times for fish and game movements.

DataSport, Inc., a Hopkins, Minnesota-based computer research firm, and Vexilar, Inc., a Minneapolis marine electronics operation, develop the forecast for the show, which is marketed on TV stations throughout the U.S.

User-Oriented Information

Now potential buyers of data-processing products can receive specific information about registered products by directly contacting current users of the hardware or software. The Association of Computer Users (ACU) has originated a new reference service that benefits both users and suppliers.

When a DP supplier registers one or more of its products with ACU, it provides at least ten references per product from current users who have agreed to answer questions from potential buyers. The potential buyer requiring specific information simply contacts ACU, which mails him a list of references on file for the product in question, and he contacts the references directly.

ACU President, Hillel Segal, notes that his company does not test or rate products. Its function is to facilitate direct communication between current and potential users of the huge range of data-processing products. Says Segal, "The companies who register products with us believe in the quality of their products, and they know it's to their advantage to supply potential customers with as much useful information as possible."

DEALER DIRECTORY

Shop the store nearest you!

Los Angeles, CA

Featuring: PolyMorphic, North Star, Imsai, Cromemco, Extensys, Speechlab products and Poly-88 Users Group software exchange. All products, 10-20% off list. We won't be undersold! **A-A-A-A Discount Computer How's, 1477 Barrington, Suite 17, Los Angeles, CA 90025, 477-8478.**

San Jose, CA

Bay area's newest computer store. Featuring the new Texas Instruments TI 99/4 home & business computer. Software for TRS-80, Apple, PET, etc. Magazines. **Hobbi-Tronics, 1378 S. Bascom Ave., San Jose, CA 95128, 998-1103.**

Sarasota, FL

Dynabyte computer systems, Hazeltine and NEC, Word-Star, Structured Systems accounting. Consulting, training, sales, service. **Glisco, Inc., 4001 Roberts Point Rd., Sarasota, FL 33581, 349-0200.**

Venice, FL

Discount prices & professional service: Cromemco, North Star, Vector Graphic, DEC, TI, Thinker Toys, Intertube, Soroc, Centronics, NEC, Selectric interfaces, Microdasys. Complete business & medical billing software available. MicroAge & Serendipity software discounted. **Sara-Tech Electronics, Inc., Computer Division, PO Box 692, Venice, FL 33595, 485-3559.**

Aurora, IL

Microcomputer systems for home or business; peripherals, software, books & magazines. Apple, Hewlett-Packard, North Star, Cromemco systems. IDS-440G printer w/Apple graphics, New HP-85 & HP calculators. **Farnsworth Computer Center, 1891 N. Farnsworth Ave., Aurora, IL 60505, 851-3888.**

Chicago, IL

Computer Hardware/Software Specialists for home and business. Largest selection of computer books, magazines and copyrighted software in Chicago Metro area. Experienced factory-trained service department. Feature Apple and Alpha Microsystems and accessories. **Data Domain of Schaumburg, 1612 E. Algonquin Road., Schaumburg, IL 60195, 397-8700.**

Chicago, IL

SS50-SS50C-6800-6809 specialists. Dealers for Gimix, Smoke Signal Broadcasting, SWTPC, TSC. Business Basic, Fast BASIC, payroll, editor-text processor. Custom programming. **AAA Chicago Computer Center, 120 Chestnut Lane, Wheeling, IL 60090, 459-0450.**

Naperville, IL

Computer systems design, programming and consultation by computer experts. Dealer for SSM, Integrand, Tarbell, Ithaca Intersystems, Verbatim, Diablo and others. Discount prices on many items. **Wilcox Enterprises, 25W178-39th St., Naperville, IL 60540, 420-8601.**

Dealers: Listings are **$15** per month in prepaid quarterly payments, or one yearly payment of **$150**, also prepaid. Ads include 25 words describing your products and services plus your company name, address and phone. (No area codes or merchandise prices, please.) Call Marcia at 603-924-7138 or write *Kilobaud MICROCOMPUTING*, Ad Department, Peterborough NH 03458.

Laurel, MD

Apple & Exidy Sorcerer. Authorized dealer for both. We also sell subscriptions to "The Source" timeshare service. Plus full line of ham equipment. **The Comm Center, Laurel Plaza, Rte. 198, Laurel, MD 20810, 792-0600.**

Worcester, MA

Complete business and personal systems on Data General, microNova and Apple. Software, books, magazines, accessories and supplies. Custom business programming available. Authorized Apple service. Downtown. **The Computer Place Inc., 11 Harvard St., Worcester, MA 01609, 755-5387.**

Garden City, MI

Complete systems for business, professional and personal applications. Custom programming available. Apple II, North Star, Vector Graphic and other lines of microcomputers, software, books, components. **Computer Center, 28251 Ford Rd., Garden City, MI 48135, 422-2570.**

Minneapolis/St. Paul, MN

Discount systems, terminals, software, peripherals & furniture for businesses, O.E.M.s, institutions & hobbyists. Send for mail-order catalog. Grand openings in two northern metro-Minneapolis locations. Z8000 systems soon! **Binary Information Systems, Box 326K, Anoka, MN 55303, 757-5878.**

St. Louis, MO

Experimenters' Paradise. Electronic and mechanical components. Computer People, Audio People, Hams, Robot Builders, Experimenters. Open six days a week. **Gateway Electronics Corp., 8123-25 Page Blvd., St. Louis, MO 63130, 427-6116.**

Akron, OH

We've got it all. Business systems. Personal systems. Software packages. Custom programming. Terminals. Printers. Service and books. Easy freeway access. 10 AM to 6 PM Monday–Saturday. **The Basic Computer Shop, Fairlawn Plaza, 2671 West Market St., Akron, OH 44313, 867-0808.**

Portland, OR

Ohio Scientific specialists for business and personal computers. Local service. Terminals, printers, custom programming. Full OSI product line on display! 10AM to 6PM M–F. **Fial Computer, 11266 SF 21st Ave., Milwaukie, OR 97222, 654-9574.**

Kingston, PA

We support Level II and Model II. Books, magazines, programs, parts, accessories, peripherals, free literature, free seminars, cassettes, floppies, filters, transformers, caps, chips, CRTs. **Artco Electronics, 302 Wyoming Ave., Kingston, PA 18704, 287-1014.**

Philadelphia/So. Jersey

North Star, fast delivery possible; word-processing systems, Heath terminals, Intertube II, immediate delivery. Free video terminal comparison, Intertec's SuperBrain, all Centronics printers, Omnitec data modems. **L & S Distributors, 44 So. Locust, Marlton, NJ 08053, 983-7444.**

Sara, Mexico

Learn how to utilize and program the Z-80 microprocessor to maintain your company's records in top shape. This technique has just been introduced to Mexico. Courses, applications, maintenance, service. **Digitales, S.A. de C.V. Sara 4612, Mexico 14, D.F., 5-17-41-59.**

Subscription Problem?

Kilobaud Microcomputing does not keep subscription records on the premises, therefore calling us only adds time and doesn't solve the problem.

Please send a description of the problem and your most recent address label to:

Kilobaud Microcomputing
Subscription Dept.
PO Box 997
Farmingdale, NY 11737

Thank you and enjoy your subscription.

CALENDAR

Personal Computer Arts Festival

On Aug. 23-24, the Philadelphia area Computer Society is sponsoring PCAF '80, a festival of talks and papers, films and graphics, demonstrations and performances. Computer musicians and artists are invited to speak, exhibit or perform: send a one-half page description, and include a tape of music, by July 1. This is being held in conjunction with the Personal Computing '80 Show at the Philadelphia Civic Center.

For information, write: PCAF '80, c/o Phila. area Computer Society, Box 1954, Phila., PA 19105.

Super Scarafest '80

Super Scarafest '80, an amateur radio and computer festival, will be held Aug. 16-17, 1980, at the Ramada Inn in North Haven, CT. Sponsored by the South Central Connecticut Amateur Radio Assoc., this event will include exhibitor booths, a ham and computer flea market and an auction.

For information, write: Super Scarafest '80, PO Box 5265, Hamden, CT 06518

Eighth World Computer Congress

Australia will co-host with Japan the Eighth World Computer Congress scheduled for Oct. 6–17, 1980, and conducted by the International Federation for Information Processing (IFIP). The congress commences in Tokyo, Japan Oct. 6–9 and concludes in Melbourne, Australia Oct. 14–17. An exhibition of hardware and related services together with submission of papers and discussions will be scheduled in each location. A single registration fee will cover attendance at both locations.

For information, contact: The Eighth World Computer Congress, IFIP Congress '80, GPO Box 880G, Melbourne, Victoria, Australia 3001.

Technical Education Courses

Integrated Computer Systems has announced its line-up of technical education courses. Each course is four days and each will be held in seven cities across the country. The courses offered are: Hands-on Microprocessor Troubleshooting, Microprocessor Hands-on Workshop, Voice Input/Output for Computers, Structured Programming, Pascal Hands-on Workshop, Software/Hardware Project Management, Computer Communication Networks, Interactive Computer Graphics and Digital Image Processing.

For information contact: Integrated Computer Systems, 3304 Pico Blvd., PO Box 5339, Santa Monica, CA 90405; or Integrated Computer Systems, 300 N. Washington St., Suite 103, Alexandria, VA 22314.

Computer Sales Workshops

Datasearch is going to be holding one-day Computer Sales Workshops for front line salespeople. The session covers such topics as qualifying your prospects, making close-mouthed prospects talk, sales presentations, organizing your time and territory and handling objections. Besides the sessions in 26 cities, in-house seminars are also available.

For schedule and details, contact Cathy Koziel, Datasearch Inc., 4954 William Arnold, Memphis, TN 38117, (901) 761-9090.

Please have calendar announcements in our hands at least three months before the month of the issue in which you want the announcement to appear. Keep announcements short and to the point, and send them to the attention of the managing editor to assure their finding their way to the editorial department. Thank you.

Four Tutorials Precede Compcon Fall '80

Four pre-conference tutorials will be presented on Monday, Sept. 22, 1980, immediately preceding Compcon Fall '80, sponsored by the IEEE Computer Society. "Distributed Processing," the theme of the conference, will be the unifying thread of the tutorials. Topics to be presented are: "Local Computer Networks," "An Overview of Distributed Processing," "Communication Technology in the 80's" and "Distributed System Design." Compcon Fall '80 is September 22–25, 1980, at the Capital Hilton Hotel, Washington, DC.

For more information, write: Compcon Fall '80, PO Box 639, Silver Spring, MD 20901, or call (301) 589-3386.

Training Management Workshops

Advanced Systems, Inc. is sponsoring a series of four-day seminars in 13 U.S. and Canadian cities. Entitled "Managing the Training Function —a Workshop," the series will begin in Chicago in May, be in New York in June and be followed by workshops in Washington, D.C., Toronto, Minneapolis, Philadelphia, Seattle, Cincinnati, Boston, St. Louis, Dallas, Houston and Atlanta.

For more information write to Advanced Systems Inc., 1601 Tonne Rd., Elk Grove Village, IL 60007.

CLASSIFIEDS

Classified advertisements are intended for use by persons desiring to buy, sell or trade used computer equipment. No commercial ads are accepted.

Two sizes of ads are available. The $5 box allows up to 5 lines of about 35 characters per line, including spaces and punctuation. The $10 box allows up to 10 lines. Minimize use of capital letters to save space. No special layouts allowed. Payment is required in advance with ad copy. We cannot bill or accept credit.

Advertising text and payment must reach us 60 days in advance of publication (i.e., copy for March issue, mailed in February, must be here by Jan. 1). The publisher reserves the right to refuse questionable or inapplicable advertisements. Mail copy with payment to: **Classifieds, Kilobaud Microcomputing, Peterborough NH 03458.** Do not include any other material with your ad as it may be delayed.

TRS-80 Exp. Int., 16K; $350. Comprint 912GP Printer; $489. 16K RAM; $50. Centronics 730 printer; $625. Compumax software; $60. Richard Smith, 3648 Madrid Dr., San Jose, CA 95132. (408) 946-0735.

Apple II, 16K; $979. Disc II; $414. 16K RAM; $50. Comprint 912S; $514. 2 sets of 4K RAM; $15 ea. Both, $25. Richard Smith, 3648 Madrid Dr., San Jose, CA 95132. (408) 946-0735.

For Sale: KIM-1 with TVT 6-5/8, power supply, 2K RAM, unencoded keyboard. Asking $175 or best offer. Call (701) 968-4525, or write T. Cartwright, Box 301, Cando, ND 58324.

For Sale: SD Expandoram, S100, 16K, with 4115s, assembled, never used. $200 or best offer. Call (701) 968-4525, or write T. Cartwright, Box 301, Cando ND 58324.

For Sale: *Kilobaud/Microcomputing* issues #1 to the present. Make offer, you ship. Call (701) 968-4525, or write T. Cartwright, Box 301, Cando, ND 58324.

Sell—Intertec Intertube computer terminal, latest version; cap 10 terminal extended keyboard, like new, teletype rolls, 7/8" wide, 8" diameter rolls, new, cheap by the case. Vic, 325 Wilson Ave., Westwood, NJ 07675.

For Sale: D. C. Hayes Micromodem for Apple II. Almost new, works fine! Only $320. Contact: H. Rothman, 218 Huntington Road, Bridgeport, CT 06608. (203) 579-0472.

For Sale: SWTPC 6800 computer, 32K RAM, SSB disk, partially assembled H14 printer and H9 CRT terminal, take all, 1800. Robert Starinsky, (312) 472-3382, 3016 N. Hamilton, Chicago, IL 60618.

For Sale: TRS-80 Model II w/32K, 1 8" floppy disk. 15% off list price of $3450. Asking $2932. Like new. Call nights, David (617) 544-7347.

New bare drives, Shugart—35TK minifloppy, MPI—40TK, Pertec—40TK, $259 each. New Base-2 printers—includes all options, $584 each. PO Box 1044, Orange, CA 92668. Call Paul Collin, (714) 847-3622.

For Sale: S-100 computer, quality APL mainframe, Vector Graphic Video Board, 32K dynamic, 8K static memory, 4 MHz Z-80 CPU board, PROM programming board, North Star disk, much software. D. Kirkeby, 5500 Cascade Way, #L, Buena Park, CA 90620. (714) 521-3437. Work: (714) 732-3355. Sell all or part.

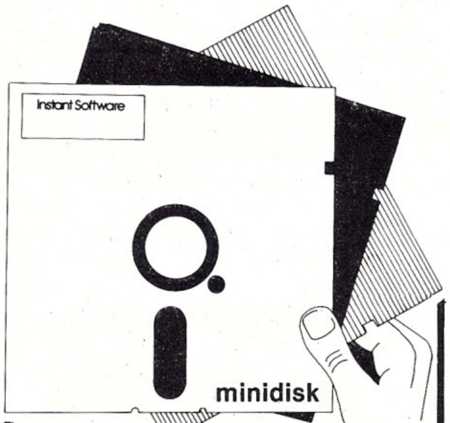
For Sale: Heath H8 8K memory boards. Four at $110 each ppd. Removed due to memory expansion. Louis Berry, 2259 N. Marter Ct., Simi Valley, CA 93065. (805) 522-3599, eves.

Wanted: Software and/or accessories for Video Brain 101 microcomputer. Most interested in Super Basic cartridge. Will pay cash. Randy Jenne, 11170-118 St., Edmonton, Alberta, Canada T5G 2W7.

Use the Classifieds

Fine Tools For Skilled Programmers

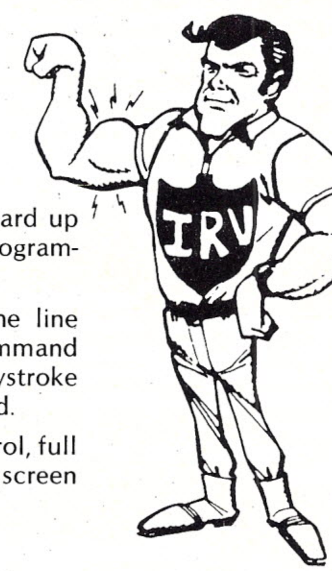
Three Utility Packages For Your TRS-80*



FOR THE TRS-80* 16K, 32K OR 48K.

PROGRAMMING POWER!

Let IRV put power in your keyboard



Input shorthand: programmable keyboard up to 255 characters per key plus a pre-programmed command set.

Relocate lines by simply changing the line number. Merge lines and relocate command blocks in the listing with simple one keystroke routine plus the Level II EDIT command.

Video screen editor has full cursor control, full power over anything that appears on the screen including line listings and graphics.

On cassette (DOS compatible).

Order No.0250R $24.95.

Disk Editor

Disk Editor is a powerful machine-language utility program that will allow you total access to ANY byte of information in ANY sector in ANY track of your diskettes. It is a fast, simple, and efficient method of modifying files, whether BASIC program, system programs, or just data. All commands are readily accessible, with no need to refer to a command table.

With Disk Editor you can examine, alter, add, and delete information with ease. Information can be retrieved from the disk by supplying track and sector information, or by giving the filespec. You can even search the disk for a specific string of characters (up to 8 characters long).

If you need hardcopy, use the LINEPRINT command to send a copy of the video display to your lineprinter.

You can transfer command from Disk Editor to Radio Shack's DEBUG and back, allowing dynamic debugging of disk I/O procedures.

Disk Editor is compatible with TRSDOS 2.1, 2.2, and 2.3, as well as with Apparat's NEWDOS. It is even capable of reading disks made by Percom's MicroDOS.

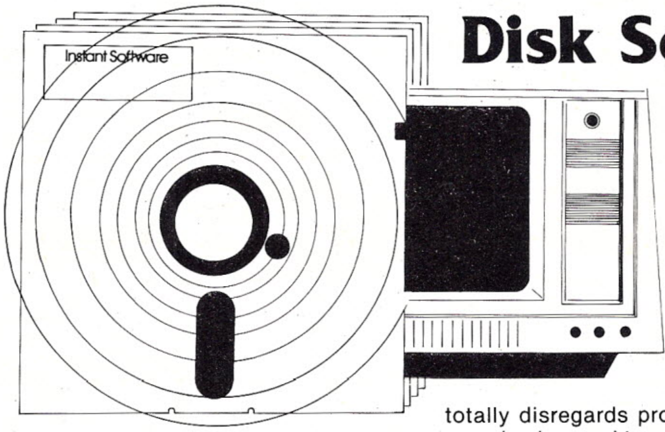
There are two versions of Disk Editor; one is for a 35 track DOS, and the other is for a 40 track DOS. Both are included in this package.

This package requires the following minimum system:

1. A TRS-80 Level II with 16K RAM.
2. An Expansion Interface.
3. A single Disk Drive.
4. Any compatible Disk Operating System. (Disk Editor is not compatible with VTOS 3.0.)

Pkg. 0180RD (disk-based version) $39.95.

Disk Scope



- **Fileloc**
- **CDisk**
- **Password**

Need to check out a disk? Perhaps you want to see how the files are stored, or you forgot your password. No problem! You've got Disk Scope.

If you know the name of the file, the **Fileloc** program will show you what tracks and sectors on the disk contain that file, as well as how much memory the file takes when loaded into RAM. This works for both program and data files. **Fileloc** then allows you to print the information, restart the program or exit to BASIC. The information obtained allows you to use the **Cdisk**program effectively.

Cdisk is a powerful little BASIC utility and test program. It will allow you to view any track and sector on your disks in ASCII, Hex and screen POKEs. It totally disregards protection codes. It can also be used to randomly check all 350 sectors of your disk for read errors.

You don't know the whole file name if you haven't got the password, so the **Password** program has been included in the Disk Scope package. This machine-language utility not only gives you a password for files, but for whole disks as well.

Whether you're a novice or a pro, if you use a disk system, you need Disk Scope.

This package requires the following minimum system:

1. A TRS-80 Level II with 16K RAM.
2. An Expansion Interface.
3. A single disk-drive.
4. Any compatible Disk Operating System.

Pkg. 0139RD (disk-based version) $19.95

*A trademark of Tandy Corporation

Prices subject to change without notice.

Instant Software™ Inc. PETERBOROUGH, N.H. 03458
603-924-7296

Ask for Instant Software at a computer store near you.

Alabama
Anderson Computers
3156 University Dr., Huntsville
Computerland of Huntsville
3020 University Dr., Huntsville
Olensky Bros.
3763 Airport Blvd., Mobile

Arizona
Ham Shack
4506-A N. 16th St., Phoenix
Millets TV & Radio
621 East Broadway, Mesa

California
AMCO Elect. Supply
635 E. Arrow Hwy., Azusa
Byte Shop
8038 Clairmont Mesa Blvd., San Diego
Byte Shop
123 E. Yorba Linda, Placentia
Byte Shop of Mt. View
1415 West El Camino Real, Mt. View
Byte Shop of Sacramento
6041 Greenback Ln., Citrus Heights
Byte Shop of San Francisco
1032 Irving St., San Francisco
Capital Computer Systems
3396 El Camino Ave., Sacramento
Computers Made Easy
819 East Ave. Q-9, Palmdale
Computer Store of San Leandro
701 MacArthur Blvd., San Leandro
Computer World
6791 Westminster Ave., Westminster
Computerland
16720 S. Hawthorne, Lawndale
Computerland of W. LA
6840 La Cienega Blvd., Inglewood
Coast Electronics
3118 No. Main St., Morro Bay
Computerland
24001 via Fabricante No 904, Mission Viejo
Hobbi-tronics
1378 So. Bascom Ave., San Jose
Hobby World
19511 Business Ctr. Dr., Unit 6 Borthridge
Huntington Computing
2020 Charles St., Corcoran
I.C.E. House Inc.
398 North E. St., San Bernardino
Jade Computer Products
4901 W. Rosecrans, Hawthorne
Malibu Microcomputing
23910A Deville Way, Malibu
Marfam Co.
6351 Almaden Rd., San Jose
Opamp/Technical Books
1033 N. Sycamore Ave., Los Angeles
PC Computers
10166 San Pablo Ave., El Cerrito
Q.I. Computers, Inc.
15818 Hawthorne Blvd., Lawndale
Radio Shack Dealer
8250 Mira Mesa Blvd., San Diego
Radio Shack Dealer
50 N. Cabrillo Hwy., Half Moon Bay
Santa Rosa Computer Center
604 7th St., Santa Rosa
Silver Spur Elect. Comm.
13552 Central Ave., Chino
The Computer Store
820 Broadway, Santa Monica

Colorado
Byte Shop
3464 S. Acoma St., Englewood
Colorado Computer Systems
311 W. 74th Ave., Westminster
Computerland of North Denver
8749 Wadsworth Blvd., Arvada
Computer Shack
1635 South Prairie, Pueblo
The Computer Store
2300 Welton St., Denver

Connecticut
American Business Computers
454 Thames St., Groton
Computerlab
130 Jefferson, New London
Computerland
1700 Post Rd., Fairfield
Computerland
60 Skiff St., Hamden
Computer Works
1439 Post Rd. E., Liberty Plaza, Westport

D.C.
The Program Store
4200 Wisconsin Ave., N.W., Washington, D.C.

Florida
Adventure International
200 Bald Cypress Ct., Longwood
AMF Electronics
11146 N. 30th St., Tampa
Boyd-Ebert Corporation
1328 West 15th St., Panama City
Computer Center
6578 Central Ave., St. Petersburg
Computer Junction
5450 So. State Rd. 7, Ft. Lauderdale
Computerland
7374 S. Tamiami Trail, Sarasota
Computerland of Ft. Lauderdale
3963 N. Federal Hwy., Ft. Lauderdale
Computerland of Jacksonville
2777-6 University Blvd. W. Jacksonville
Computerland of Tampa
1520 E. Fowler Ave., Tampa
Computerland of West Palm Beach
4275 Okeechobee Blvd., West Palm Beach
Computer Shack
3336 Beach Blvd., Jacksonville
Computer System Resources Inc.
3222 S.W. 35th Blvd., Gainesville
Curtis Waters Enterprises
236 Talbot Ave., Melbourne
Heath Kit Electronic
4705 W. 16th Ave. Center, Hialeah
HIS Computermation
1295 Cypress Ave., Melbourne
Ukatan Computer Store
Airport Rd., Destin
Williams Radio & TV Inc.
2062 Liberty St., Jacksonville
Your Basic Computer Store
971 Seaway Dr., Ft. Pierce

Georgia
Atlanta Computer Mart
Atlanta
Computerland of Atlanta
2423 Cobb Parkway, Smyrna

Hawaii
Computerland of Hawaii
567 N. Federal Hwy., Honolulu
Radio Shack Assoc. Store
1712 S. King St., Honolulu

Idaho
Electronic Specialists
8411 Fairview Ave., Boise

Illinois
Computerland
4507 North Sterling, Peoria
Computerland
9511 N. Milwaukee Ave., Niles
Computer Station
3659 Nameoki Rd., Granite City
Garcia & Associates
203 No. Wabash Ave., Suite 1510, Chicago
Midwest Micro Computers, Inc.
708 S. Main St., Lombard

Indiana
Computer Center of South Bend
51591 US 31 North, South Bend
Data Domain
221 W. Dodds, Bloomington
Fall Creek Electronics Store
732 Center St., Pendleton

Iowa
Memory Bank
4128 Brady St., Davenport

Kansas
Central Kansas Computers
6 S. Broadway, Herington

Louisiana
Computer Shoppe Inc.
3225 Danny Park, Suite 222, Metairie

Maine
Main Computronics
Intown Plaza, Bangor
Radio Shack
315 Main Mall Rd., So. Portland

Maryland
Jack Fives Electronics
4608 Debilen Circle, Pikesville
The Comm Center
9624 Ft. Meade Rd., Laurel

Massachusetts
ComputerCity
175 Main St., Charlestown
ComputerCity
50 Worcester Rd., Framingham
Computerland of Boston
214 Worcester Rd., Wellesley
Computer Packages Unlimited
244 W. Boylston St., West Boylston
Lighthouse Computer Software
14 Fall River Ave., Rehobath
New England Electronics Co.
679 Highland Ave., Needham
Small Business System Group
Main St., Dunstable
The Computer Store
120 Cambridge St., Burlington

Michigan
Computer Center
28251 Ford Rd., Garden City
Computer Connections
38437 Grand River, Farmington Hills
Computerland of Grand Rapids
2927 28th St. S.E., Kentwood
Computerland of Southfield
29673 Northwestern Hwy., Southfield
Computer Mart
560 W. 14 Mile Rd., Clawson
Computer Room
455 E. Michigan Ave., Kalamazoo
Computronix Corp.
423 S. Saginaw Rd., Midland
Hobby House
1035 W. Territorial Rd., Battle Creek
Main Systems Inc.
1161 No. Ballenger Hwy., Flint
The Alternate Source
1806 Ada, Lansing
The Eight Bit Corner
722 Evanston Ave., Muskegon
Ye Olde Teacher Shoppe
1823 Witmyre St., Ypsilanti

Minnesota
Computerland of Hopkins
11319 Hwy F., Hopkins
Digital Den
Burnsville Center
Minnesota Software Inc.
5422 Fisher St., White Bear Lake
Zim Computers
5717 Xerxes Ave., N. Brooklin Center

Mississippi
Dyer's, Inc.
200 E. Main St., West Point
Softwarehouse
816 Foley St., Jackson
W. Vernon Foster Inc.
816 Foley St., Jackson

Missouri
Computervan, Inc.
51 Florissant Oaks Shopping Center Florissant
Software Shack
16501 Greenwald Court, Belton

Nebraska
Computerland of Omaha
11031 Elm St., Omaha
Midwest Computer Co. Inc.
8625 I St., Omaha
Midwest Computer Co. Inc.
4442 S. 84th St., Omaha
Midwest Computer Co. Inc.
4403 S. 87th St., Omaha
Scottsbluff Typewriters Inc.
1824 Broadway, Scottsbluff

Nevada
Century 23
4566 Spring Mountain Rd., Las Vegas

New Hampshire
Bitsnbytes Computer Center
568 Pleasant St., Concord
ComputerCity
1525 S. Willow, Manchester
Paul's TV
Main St., Fremont
Portsmouth Computer Center
31 Raynes Ave., Portsmouth
Radio Shack Assoc. Store
Fairbanks Plaza, Keene

New Jersey
Computer Corner of NJ
439 Rte. #23, Pompton Plains
Computer Encounter
2 Nassau St., Princeton
Computerland
35 Plaza Rte. #4, W. Paramus
Computer Mart of NJ
501 Rte. 27, Iselin
Crowley's
Rd. #3, Whitehouse Station
Dave's Electronics
Pennsville Shopping Ctr., Pennsville
GHB Enterprises Inc.
Rte. 38, Rudderaw Ave., Mapleshade
Lashen Electronics Inc.
21 Broadway, Denville

New Mexico
Autel Electronics Co.
232 Wisconsin N.E., Albuquerque
South West Computer Center
121 Wyatt Drive, Suite 7, Las Cruces
Thomas E. Carr Jeweler
1300A Tenth St., Alamogordo

New York
Aristo Craft
314 Fifth Ave., NYC
Bits & Bytes
2800 Straight Rd., Fredonia
Computer Corner
200 Hamilton Ave., White Plains
Computer Era Corp.
1570 3rd Ave., New York
Computer Factory
485 Lexington Ave., NYC
Computer House, Inc.
721 Atlantic Ave., Rochester
Computerland of Nassau
79 Westbury Ave., Carle Place
A World of Computers
519 Boston Post Rd., Port Chester
Comtek Electronics, Inc.
2666 Coney Island Ave., Brooklyn
Comtek Electronics, Inc.
Staten Island Mall
Store 220A, Staten Island
Digibyte Systems Corp.
31E. 31st St., New York
80-Microcomputer Services
118 Masten Ave., Cohoes
Home Computer Center
671 Monroe Ave., Rochester
Key Electronics
Schenectady
Mr. Computer
Imp. Plaza, Rte. 9, Wappingers Falls
Softron Systems
308 Columbia Turnpike, Rensselaer
The Computer Tree Inc.
409 Hooper Rd., Endwell
Upstate Computer Shop
629 French Rd., Campus Plaza New Hartford

North Carolina
Byte Shop of Raleigh
1213 Hillsborough St., Raleigh

Ohio
Altair Business Systems, Inc.
5252 North Dixie Dr., Dayton
Astro Video Electronics
504 E. Main St., Lancaster
Cincinnati Computer Store
4816 Interstate Dr., Cincinnati
Computerland
4579 Great Northern Blvd., N. Olmstead
Computerland
6429 Busch Blvd., Columbus
Computerland
1288 Som Rd., Mayfield Heights
Computer Store of Toledo
18 Hillwyck Dr., Toledo
Forbees Microsystems Inc.
35 N. Broad, Fairborn
Microcomputer Center
7900 Paragon Rd., Dayton
Micro-Mini Computer World
74 Robinwood, Columbus
21st Century Shop
16 Convention Way, Cincinnati
Universal Amateur Radio, Inc.
1280 Aida Dr., Columbus

Oklahoma
Sounds, Etc.
Hyw. 33, Watonga
Vern Street Products
Radio Shack Dealer
114 W. Taft St., Sapulpa

Oregon
Computerland of Portland
12020 S.W. Main St., Tigard
Computer Pathways Unlimited, Inc.
2151 Davcor St. S.E., Salem
TRS-80 Products Ltd.
3520 S.E. Vineyard Rd., Portland

Pennsylvania
Artco Elect.
302 Wyoming Ave., Kingston
Artco Elect.
Back Mountain Shop. Ctr. Shavertown
Audio Mart
518 Fifth Ave., New Brighton
Computer Workshoppe
3848 William Penn Hwy, Monroeville
Computerland of Harrisburg
4644 Carlisle Pike, Mechanicsburg
Computerland of Pittsburgh
5499 William Flynn Hwy., Gibsonia
Erie Computer Co.
2127 West 8th St., Erie
Mighty Byte Computer Center
537 Easton Rd., Horsham
Personal Computer Corp.
24-26 West Lancaster Ave., Paoli
Personal Computer Corp.
Frazer Mall, Lancaster Ave., Frazer

Rhode Island
Computer City
165 Angell St., Providence
Digital World, Inc.
329 Bald Hill Rd., Warwick

South Dakota
CB Radio Shack
21st and Broadway, Yankton

Tennessee
ACS
1100 8th Ave. So., Nashville
Computerlab
671 S. Menden Hall Rd., Memphis
H & H Electronics Inc.
509 N. Jackson St., Tullahoma

Texas
Computer Port
2142 N. Collins, Arlington
Houston Computer Tech
5313 Bissonet, Bellarie
Interactive Computer
7620 Dashwood, Houston
K.A. Elect.
9090 Stemmons Frwy., Dallas
Pan American Elect. Inc.
1117 Conway, Mission
Radio Shack Dealer
21969 Katy Freeway, Katy
The Compute Shop
6353 Camp Bowie Blvd., Ft. Worth
Waghalter Books Inc.
3 Greenway Plaza E., Houston

Utah
DC Computer Co.
1911 West 70 South, Provo
Quality Technology
470 E. 2nd So., Salt Lake City

Virginia
Computer Works
Rte. 6, Box 65A, Harrisonburg
Home Computer Center
2927 Virginia Beach Blvd. Virginia Beach
Southside Radio Comm.
135 Pickwick Ave., Colonial Heights

Washington
American Mercantile Co. Inc.
2418 1st Ave. S., Seattle
Byte Shop of Bellevue
14701 N.E. 20th St., Bellevue
Computer Connection Inc.
3100 NW Bucklin Hill Rd., Silverdale
Computerland of South King Co.
1500 S. 336 St., Suite 12 Federal Way
Personal Computers
S 104 Freva, Spokane
Ye Old Computer Shop
1301 G. Washington, Richland

West Virginia
The Computer Corner Inc.
22 Beechurst Ave., Morgantown
The Computer Store
Municipal Parking Bldg., Charleston

Wisconsin
Byte Shop Of Milwaukee
6019 West Layton Ave., Greenfield
Computerland
690 S. Whitney Way, Madison
Computerworld
3015 W. Wisconsin Ave., Appleton
Petted Microsystems
4265 W. Loomis Rd., Milwaukee

Wyoming
Computer Concepts
1104 Logan Ave., Cheyenne

Puerto Rico
The Microcomputer Store
1568 Ave. Jesus T. Pinero
Caparra Terrace

Canada

CANADIAN DISTRIBUTORS:
Micron Distributing
409 Queen St., W. Toronto, Ont. M5V 2A5

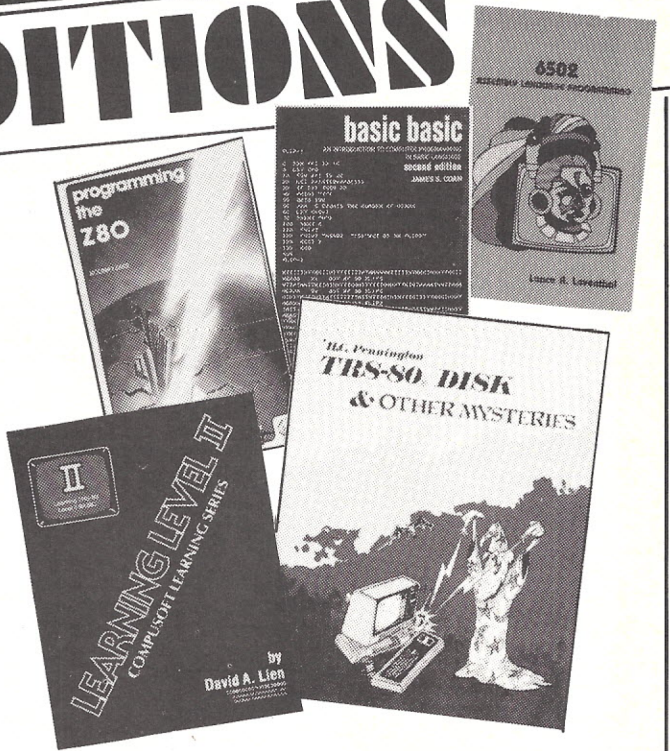
Computerland of Winnipeg
715 Portage Ave., Winnipeg, Man.
Compumart
411 Roosevelt Ave., Ottawa, Ontario
Micromatic Systems Inc.
101 8136 Park Rd., Richmond, B.C.
Micro Shack of W. Canada
333 Park Street, Regina. Sask.
Orthon Holdings Ltd.
12411 Stony Plain Road
Edmonton, Alberta
Total Computer Systems
Ajax, Ontario

Instant Software™ Inc.

PETERBOROUGH, N.H. 03458

kb microcomputing book nook

NEW ADDITIONS



● **PROGRAMMING THE Z-80** – BK1122 – by Rodnay Zaks. Here is assembly language programming for the Z-80 presented as a progressive, step-by-step course. This book is both an educational text *and* a self-contained reference book, useful to both the beginning and the experienced programmer who wish to learn about the Z-80. Exercises to test the reader are included. $14.95.*

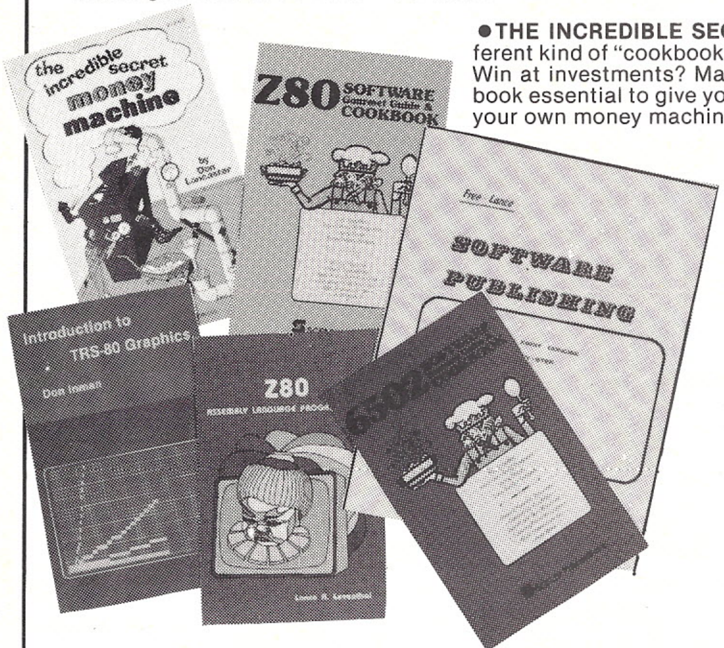
● **6502 ASSEMBLY LANGUAGE PROGRAMMING** – BK1176 – by Lance A. Leventhal. This book provides comprehensive coverage of the 6502 microprocessor assembly language. Leventhal covers over 80 programming examples from simple memory load loops to complete design projects. Features include 6502 assembler conventions, input/output devices and interfacing methods, and programming the 6502 interrupt system. $12.50.*

● **TRS-80 DISK AND OTHER MYSTERIES** – BK1181 – by Harvard C. Pennington. This is the definitive work on the TRS-80 disk system. It is full of detailed "How to" information with examples, samples and in-depth explanations suitable for beginners and professionals alike. The recovery of one lost file is worth the price alone. $22.50.*

● **BASIC BASIC (2ND EDITION)** – BK1026 – by James S. Coan. This is a textbook which incorporates the learning of computer programming using the BASIC language with the teaching of mathematics. Over 100 sample programs illustrate the techniques of the BASIC language and every section is followed by practical problems. This second edition covers character string handling and the use of data files. $9.45.*

● **LEARNING LEVEL II** – BK1175 – by David Lien. Written especially for the TRS-80, this book concentrates on Level II BASIC, exploring every important BASIC language capability. Updates are included for those who have studied the Level I User's Manual. Sections include: how to use the Editor, dual cassette operation, printers and peripheral devices, and the conversion of Level I programs to Level II. $15.95.*

● **THE INCREDIBLE SECRET MONEY MACHINE** – BK1178 – by Don Lancaster. A different kind of "cookbook" from Don Lancaster. Want to slash taxes? Get free vacations? Win at investments? Make money from something that you *like* to do? You'll find this book essential to give you the key insider details of what is really involved in starting up your own money machine. $5.95.*



● **FREELANCE SOFTWARE PUBLISHING** – BK1179 – by B. J. Korites. "This book is about money and how to make it by writing and selling computer programs," (author's foreword). If you have the skills to write a saleable program, you now need to acquire the skills to sell that program. This compact book comprehensively covers the entire publishing process and many aspects of software salesmanship. $14.95.*

● **Z-80 SOFTWARE GOURMET GUIDE AND COOKBOOK** – BK1045 – by Nat Wadsworth. Scelbi's newest cookbook! This book contains a complete description of the powerful Z-80 instruction set and a wide variety of programming information. Use the author's ingredients including routines, subroutines and short programs, choose a time-tested recipe and start cooking! $14.95.*

● **6502 SOFTWARE GOURMET GUIDE AND COOKBOOK** – BK1055 – by Robert Findley. This book introduces the BASIC language programmer into the realm of machine-language programming. The description of the 6502 structure and instruction set, various routines, subroutines and programs are the ingredients in this cookbook. "Recipes" are included to help you put together exactly the programs to suit your taste. $10.95.*

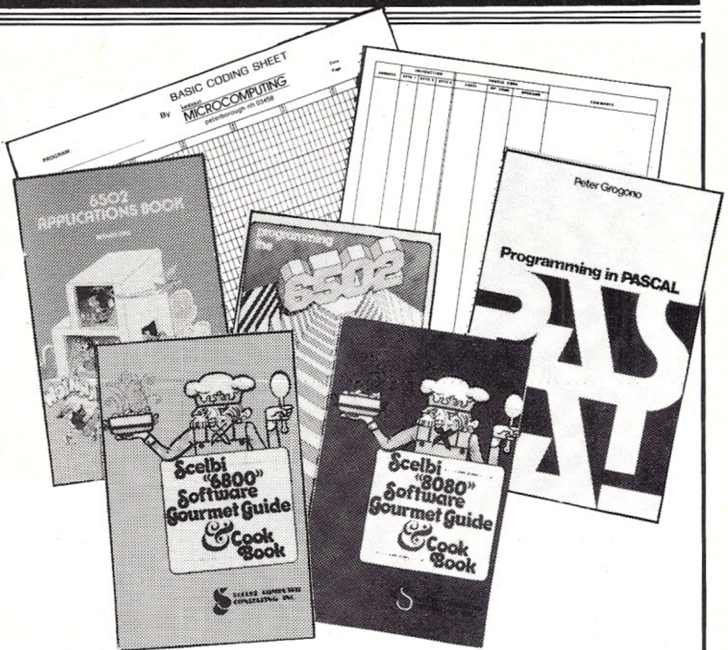
● **INTRODUCTION TO TRS-80 GRAPHICS** – BK1180 – by Don Inman. Dissatisfied with your Level I or Level II manual's coverage of graphics capabilities? This well-structured book (suitable for classroom use) is ideal for those who want to use all the graphics capabilities built into the TRS-80. A tutorial method is used with many demonstrations. It is based on the Level I, but all material is suitable for Level II use. $8.95.*

● **Z-80 ASSEMBLY LANGUAGE PROGRAMMING** – BK1177 – by Lance A. Leventhal. This book thoroughly covers the Z80 instruction set, abounding in simple programming examples which illustrate software development concepts and actual assembly language usage. Features include Z80 I/O devices and interfacing methods, assembler conventions, and comparisons with 8080A/8085 instruction sets and interrupt structure. $12.50.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to Kilobaud *Microcomputing* Book Department • Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. All orders add $1.00 handling. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

kb microcomputing book nook



● **PROGRAMMING IN PASCAL** – BK1140 – by Peter Grogono. The computer programming language PASCAL was the first language to embody in a coherent way the concepts of structured programming, which has been defined by Edsger Dijkstra and C.A.R. Hoare. As such, it is a landmark in the development of programming languages. PASCAL was developed by Niklaus Wirth in Zurich; it is derived from the language ALGOL 60 but is more powerful and easier to use. PASCAL is now widely accepted as a useful language that can be efficiently implemented, and as an excellent teaching tool. It does not assume knowledge of any other programming language; it is therefore suitable for an introductory course. $12.95.*

● **8080 SOFTWARE GOURMET GUIDE AND COOKBOOK** – BK1102 – If yu have been spending too much time developing simple routines for your 8080, try this new book by Scelbi Computing and Robert Findley. Describes sorting, searching, and many other routines for the 8080 user. $10.95.*

● **6800 SOFTWARE GOURMET GUIDE AND COOKBOOK** – BK1075 – Like its culinary cousin, *The 8080 Gourmet Guide*, this new book by Scelbi Computing and Robert Findley describes sorting, searching and other routines – this time for the 6800 user. $10.95.*

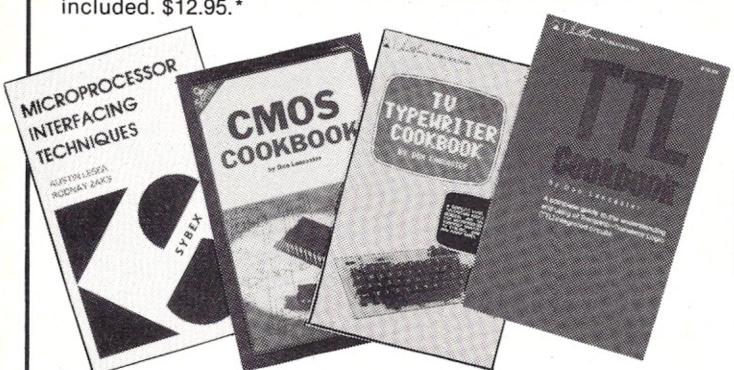
● ***MICROCOMPUTING* CODING SHEETS** *Microcomputing's* dozen or so programmers wouldn't try to work without these handy scratch pads, which help prevent the little errors that can cost hours and hours of programming time. Available for programming is **Assembly/Machine Language** (PD1001), which has columns for address, instruction (3 bytes), source code (label, op code, operand) and comments; and for **BASIC** (PD1002) which is 72 columns wide. 50 sheets to a pad. $2.39.*

● **PROGRAMMING THE 6502 (Second Edition)** – BK1005 – Rodnay Zaks has designed a self-contained text to learn programming, using the 6502. It can be used by a person who has never programmed before, and should be of value to anyone using the 6502. The many exercises will allow you to test yourself and practice the concepts presented. $12.95.*

● **6502 APPLICATIONS BOOK** – BK1006 – Rodnay Zaks presents practical-application techniques for the 6502 microprocessor, assuming an elementary knowledge of microprocessor programming. You will build and design your own domestic-use systems and peripherals. Self-test exercises included. $12.95.*

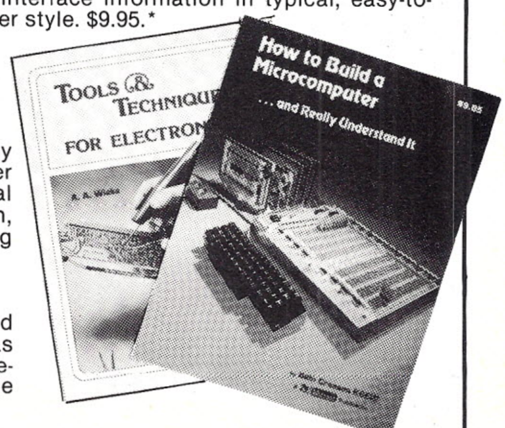
● **MICROPROCESSOR INTERFACING TECHNIQUES** – BK1037 – by Austin Lesea & Rodnay Zaks will teach you how to interconnect a complete system and interface it to all the usual peripherals. It covers hardware and software skills and techniques, including the use and design of model buses such as the IEEE 488 or S100. $15.95.*

● **TTL COOKBOOK** – BK1063 – by Donald Lancaster. Explains what TTL is, how it works, and how to use it. Discusses practical applications, such as a digital counter and display system, events counter, electronic stopwatch, digital voltmeter and a digital tachometer. $9.50.*



● **CMOS COOKBOOK** – BK1011 – by Don Lancaster. Details the application of CMOS, the low power logic family suitable for most applications presently dominated by TTL. Required reading for every serious digital experimenter! $10.50.*

● **TVT COOKBOOK** – BK1064 – by Don Lancaster. Describes the use of a standard television receiver as a microprocessor CRT terminal. Explains and describes character generation, cursor control and interface information in typical, easy-to-understand Lancaster style. $9.95.*



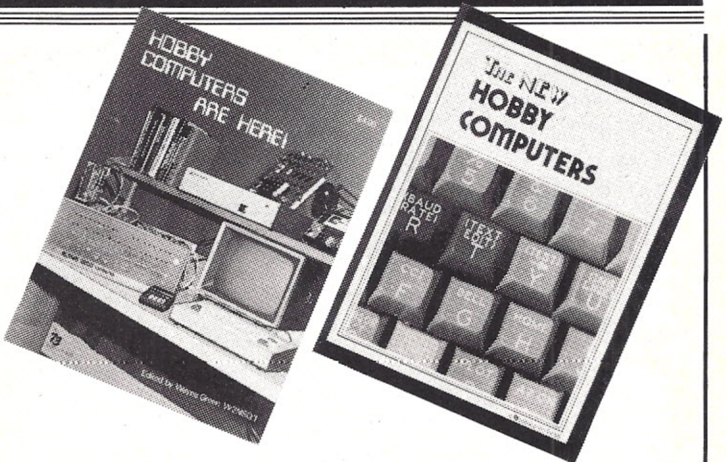
● **HOW TO BUILD A MICROCOMPUTER – AND REALLY UNDERSTAND IT** – BK7325 – by Sam Creason. The electronics hobbyist who wants to build his own microcomputer system now has a practical "How-To" guidebook. This book is a combination technical manual and programming guide that takes the hobbyist step-by-step through the design, construction, testing and debugging of a complete microcomputer system. Must reading for anyone desiring a true understanding of small computer systems. $9.95.*

● **TOOLS & TECHNIQUES FOR ELECTRONICS** – BK7348 – is an easy-to-understand book written for the beginning kit builder as well as the experienced hobbyist. It has numerous pictures and descriptions of the safe and correct ways to use basic and specialized tools for electronic projects as well as specialized metal working tools and the chemical aids which are used in repair shops. $4.95.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to Kilobaud *Microcomputing* Book Department • Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. All orders add $1.00 handling. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the above address.

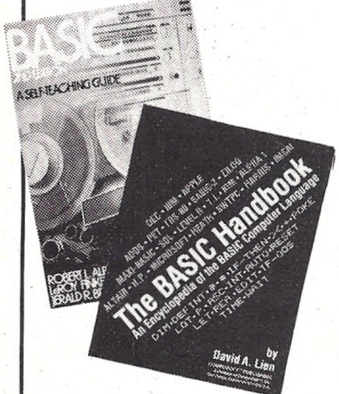
FOR TOLL FREE ORDERING CALL 1-800-258-5473

kb microcomputing book nook



●**HOBBY COMPUTERS ARE HERE!**—BK7322—If you (or a friend) want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with the fundamentals and explains the circuits, and the basics of programming. This book has the highest recommendations as a teaching aid for newcomers. $4.95.*

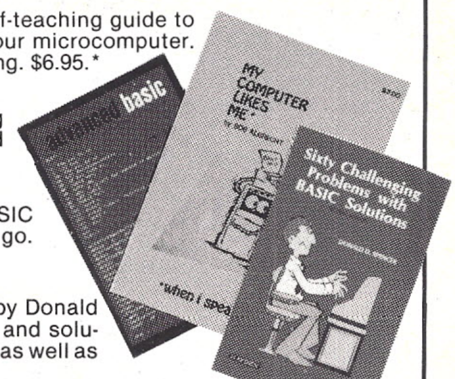
●**THE NEW HOBBY COMPUTERS**—BK7340—This book takes it from where "HOBBY COMPUTERS ARE HERE!" leaves off, with chapters on Large Scale Integration, how to choose a microprocessor chip, an introduction to programming, low cost I/O for a computer, computer arithmetic, checking memory boards . . . and much, much more! Don't miss this tremendous value! Only $4.95.*



●**THE BASIC HANDBOOK**—BK1174—by David Lien. This book is unique. It is a virtual ENCYCLOPEDIA of BASIC. While not favoring one computer over another, it explains over 250 BASIC words, how to use them and alternate strategies. If a computer does not possess the capabilities of a needed or specified word, there are often ways to accomplish the same function by using another word or combination of words. That's where the HANDBOOK comes in. It helps you get the most from your computer, be it a "bottom-of-the-line" micro or an oversized monster. $14.95.*

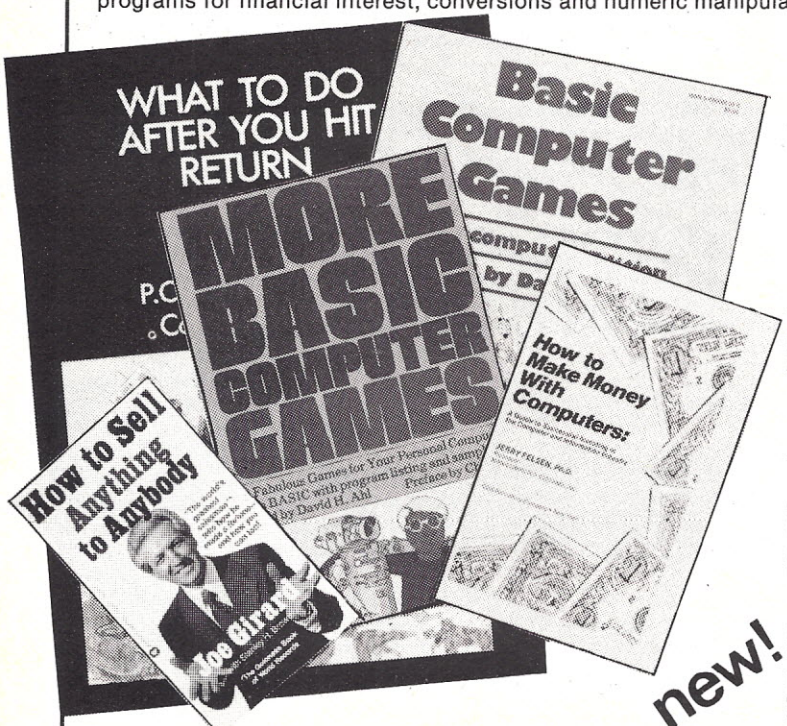
●**BASIC NEW 2ND EDITION**—BK1081—by Bob Albrecht. Self-teaching guide to the computer language you will need to know for use with your microcomputer. This is one of the easiest ways to learn computer programming. $6.95.*

●**ADVANCED BASIC**—BK1000—Applications, including strings and files, coordinate geometry, area, sequences and series, simulation, graphing and games. $9.65*.



●**MY COMPUTER LIKES ME . . . WHEN I SPEAK BASIC**—BK1039—An introduction to BASIC . . . simple enough for kids. If you want to teach BASIC to anyone quickly, this is the way to go. $2.00.*

●**SIXTY CHALLENGING PROBLEMS WITH BASIC SOLUTIONS (2nd Edition)**—BK1073—by Donald Spencer, provides the serious student of BASIC programming with interesting problems and solutions. No knowledge of math above algebra required. Includes a number of game programs, as well as programs for financial interest, conversions and numeric manipulations. $6.95.*



●**HOW TO SELL ANYTHING TO ANYBODY**—BK7306—According to *The Guinness Book of World Records*, the author, Joe Girard, is "the world's greatest salesman." This book reveals how he made a fortune—and how you can, too. $2.25.*

●**HOW TO MAKE MONEY WITH COMPUTERS**—BK1003—In 10 information-packed chapters, Jerry Felsen describes more than 30 computer-related, money-making, high profit, low capital investment opportunities. $15.00.*

●**WHAT TO DO AFTER YOU HIT RETURN**—BK1071—PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, many illustrations. Lunar Landing, Hammurabi, King, Civel 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $10.95.*

●**BASIC COMPUTER GAMES**—BK1074—Okay, so once you get your computer and are running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. $7.50.*

●**MORE BASIC COMPUTER GAMES**—BK1182—Edited by David H. Ahl. More fun in BASIC! 84 new games from the people who brought you *BASIC Computer Games* including such favorites as Minotaur (battle the mythical beast) and Eliza (unload your troubles on the doctor at bargain rates). Complete with game description, listing and sample run. $7.50.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to Kilobaud *Microcomputing* Book Department • Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. All orders add $1.00 handling. Please allow 4–6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

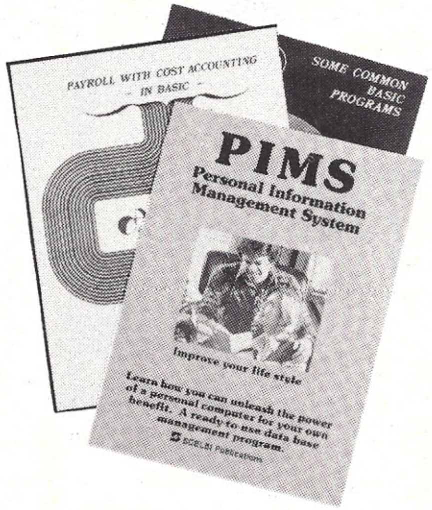
kb microcomputing book nook

●**AN INTRODUCTION TO MICROCOMPUTERS, VOL. 0**—BK1130—The Beginner's Book—Written for readers who know nothing about computers—for those who have an interest in how to use computers—and for everyone else who must live with computers and should know a little about them. The first in a series of 4 volumes, this book will explain how computers work and what they can do. Computers have become an integral part of life and society. During any given day you are affected by computers, so start learning more about them with Volume 0. $7.95.*

●**VOL. I-BK1030-2nd Edition completely revised.** Dedicated to the basic concepts of microcomputers and hardware theory. The purpose of Volume I is to give you a thorough understanding of what microcomputers are. From basic concepts (which are covered in detail), Volume I builds the necessary components of a microcomputer system. This book highlights the difference between minicomputers and microcomputers. $12.50.*

●**VOL. II**—BK1040 (with binder)—$30.00*—Contains descriptions of individual microprocessors and support devices used only with the parent microprocessor. Volume II describes all available chips.

●**VOL. III**—BK1133 (with binder)—$20.00.* Contains descriptions of all support devices that can be used with any microprocessor.



**TOLL FREE ORDERING
CALL 1-800-258-5473**

●**PAYROLL WITH COST ACCOUNTING—IN BASIC**—BK1001— by L. Poole & M. Borchers, includes program listings with remarks, descriptions, discussions of the principle behind each program, file layouts, and a complete user's manual with step-by-step instructions, flowcharts, and simple reports and CRT displays. Payroll and cost accounting features include separate payrolls for up to 10 companies, time-tested interactive data entry, easy correction of data entry errors, job costing (labor of distribution), check printing with full deduction and pay detail, and 16 different printed reports, including W-2 and 941 (in CBASIC). $20.00.*

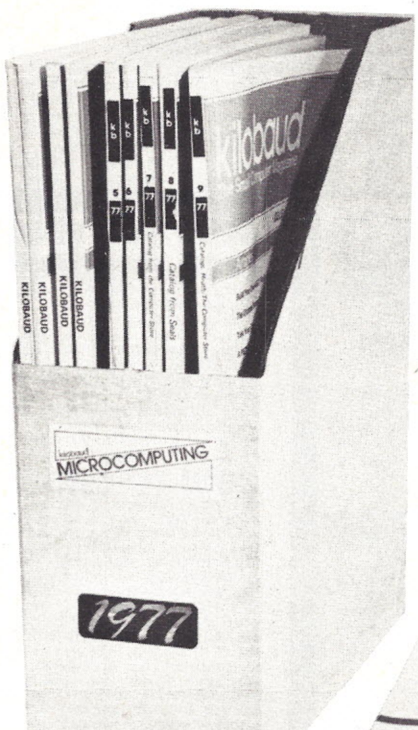
●**SOME COMMON BASIC PROGRAMS**—BK1053—published by Adam Osborne & Associates, Inc. Perfect for non-technical computerists requiring ready-to-use programs. Business programs, plus miscellaneous programs. Invaluable for the user who is not an experienced programmer. All will operate in the stand-alone mode. $12.50 paperback.*

●**PIMS: PERSONAL INFORMATION MANAGEMENT SYSTEM**—BK1009—Learn how to unleash the power of a personal computer for your own benefit in this ready-to-use data-base management program. $9.95.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to Kilobaud *Microcomputing* Book Department • Peterborough NH 03458. Be sure to include check or detailed credit card information. No C.O.D. orders accepted. All orders add $1.00 handling. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write to Customer Service at the above address.

only a few left...

1977 KILOBAUD: the COMPLETE COLLECTION for $20



That's right, we're liquidating the last issues of *Kilobaud* we have been saving . . . these are COLLECTOR'S COPIES . . . with several of the individual copies worth more than we're asking for the complete year.

This includes the very hard-to-get #1 issue which has been going for as much as $25 at auctions . . . and a library shelf box . . . the whole works, complete. When these rare copies are gone, that will be the end of it. This is your last chance to get a complete collection of Kilobaud . . . starting from the beginning.

During this first year *Kilobaud* had 224 feature articles, including 85 with complete program listings. Just as a source of programs this is a bargain.

The $20 price includes the library shelf box and shipping charges within the USA. You can call our toll free number and order this special: (800) 258-5473. Have your credit card handy.

Please specify #KB1977 when ordering.

America's One-Stop Computer Center

HOBBYWORLD® ELECTRONICS, INC.

✓10

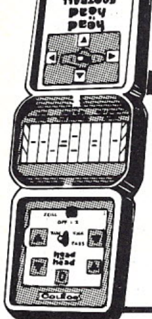
Call Toll-Free: USA (800) 423-5387
In California: (800) 382-3651
Local & Outside USA: (213) 886-9200

BETTER BUSINESS BUREAU BBB® MEMBER

MICROPOLIS® DISK DRIVES

Imagine a 5¼" floppy disk system with all the storage capacity of an 8" floppy system, and more!
Micropolis can give you more storage because they pack more data onto every disk. Ordinary 5¼" floppies provide just 35 tracks per drive and store 70 to 130 K bytes of data. Instead, Micropolis uses 77 tracks-each with 16 sectors of 156 bytes to yield an incredible storage capacity of 315k bytes per drive.
And that's not all! Reliability doesn't just happen by accident. At Micropolis reliability is engineered into each step of manufacturing. For example, most 5¼" floppy disks cut costs by using a plastic cam or cam follower to position the read/write head. Micropolis chose to use the strength and durability of an all steel cam and cam follower. Sure, it costs more, but it gives you more accurate tracking over a significantly greater lifespan which adds up to a lower cost per byte with use. Software from Micropolis includes a complete DOS, (disk operating system), and disk Extended Basic designed for 8080/Z80 microcomputer systems. The DOS is complete with an assembler, editor, file management functions and disk utilities. Micropolis BASIC is a complete, powerful programming tool for developing, testing, executing and maintaining basic programs.
The model 1043 MOD II is a single floppy disk with 315k bytes storage and includes the S-100 disk Controller board. If you need more storage, or simply want to save even more money, then order the model 1053 MOD II dual disk system with 630k bytes storage capacity and S-100 Controller board. Micropolis DOS and disk extended BASIC are standard with both units.

Cat No.	Description	Weight	Price
2563	Model 1043 mod II single drive	9 lbs	$1145.00
2564	Model 1053 mod II dual drive	18 lbs	$1895.00



"HEAD TO HEAD" ELECTRONIC FOOTBALL

Two ways to play, against the computer or against a friend, and two levels of skill, semi-pro and pro. This versatile computerized game will provide hours of fun with authentic football action. Complete offense and defense control, pass block, run and kick/punt. LED statistical display and electronic sound effects. Batteries not included. Wt. 10 oz.
Cat No. 2404 **$55.60**

SSM VB2 VIDEO BOARD

I/O controlled video interface board. With a TV monitor, the VB2 becomes a video terminal! No other I/O card is required for keyboard input and video display. The cursor, linefeed, carriage return, backspace and clear screen are hardware controlled. The display is 64 X 16, all upper case and is selectable for white-on-black or black-on-white.
Produces a clear, bright display, and features adjustable picture size and character width. Circuitry is provided to drive a speaker for beep tone.

Cat No.	Description	Price
1438	VB2 kit	$169.00
1439	VB2 A&T	$234.00

PROGRAMMA THE SOUNDING BOARD For APPLE II $129.95

Enter the world of microcomputer music and sound!! One board will turn your Apple into a fine musical instrument and, in addition, produce sound effects which will spice up any program. Each board has three programmable voices, and on-board generator and built in amplifier which can drive an 8 ohm speaker. The Sounding Board has a five octave range starting at 55hz which is below the bass clef, to 176hz which is 2 octaves above the treble clef.
The Apple will hold 6 boards which would give you the capability to create 18 simultaneously programmable voices. Music can be composed, edited, played and then stored. Musical notes can be entered directly from the keyboard. Included with your board is an interactive music editor, sample music and a demonstration program which plays continuous music.
The best board of it's type on the market today for a fraction of the cost of the other, less productive offerings. Weight 3 oz.
Cat No. 2561

TRS-80 CP/M

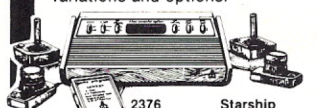
At last, CP/M is available for the TRS-80! Long a standard for software development and interchange for all the "other" 8080/Z80 computers on the market, CP/M will now provide the same environment for the TRS-80.
CP/M is a file-oriented disk operating system that provides a common set of utilities for program development and operation. There are six built in commands plus utilities called in from disk. CP/M will run on a TRS-80 with as little as 16k of memory and one disk drive. Comes complete with six manuals. CP/M is a registered trade mark of Digital Research.
Cat No. 1679
TRS-80 Level II 16k-w/disk **$149.95**

THE PIE 2.0 $79.95 (Programma Improved Editor)

Don't be mislead by the low price of this outstanding wordprocessing package. PIE 2.0 is a powerful text editor and print format processor that has all the bells and whistles expected of wordprocessing software costing three times as much. Some features include:
1) Character/line insert and delete
2) Complete Cursor mobility
3) String search forward and backward
4) Single, conditional or global search and replace
5) Move and/or copy blocks of text
6) Page scrolling
7) Tabs, margins, paragraphing, etc.
Research conducted by IBM Corp. revealed that the time required to create, edit and complete a one page document was decreased by as much as 60% when comparing the performance of a Wordprocessing system to an ordinary typewriter. Finding ways to remain competitive these days is a challenge for the business executive. Todays office can substantially improve their daily productivity level with PIE 2.0 Wordprocessing software and an Apple II computer with 32k RAM memory.
As a businessman you want every dollar you spend to count, so wordprocessing makes sense, and PIE 2.0 Wordprocessing software gives you more for your hard earned dollar. PIE 2.0 Wordprocessing software comes complete with program diskette and detailed documentation in a handsome, simulated leather binder.
Cat No. 2562

ATARI® HOME VIDEO SYSTEM

The nation's best selling home entertainment center is here! Currently supports a library of 25 video game cartridges with over 1300 variations and options. This system comes with interchangeable joystick and paddle controllers, realistic sound effects, produces bright & crisp colors on your TV screen and special circuits to protect your home TV. Also includes ATARI's "Combat" game with 108 variations and options.



CAT NO.	DESCRIPTION	WT.	PRICE
2375	ATARI Home Video System	8 lb.	$183.50
2206	Driving Controller-Pair	2 lb.	$18.59
2207	Paddle Controller-Pair	2 lb.	$18.59
2208	Joystick Controller-Pair	2 lb.	$18.59

ATARI Game Cartridges

2376	Starship	$19.50	2389	Football	$19.50
2377	Black Jack	$19.50	2390	Bowling	$19.50
2378	Space War	$12.95	2391	Skydiver	$19.50
2379	Surround	$12.95	2392	Fun With Numbers	$19.50
2380	Slot Machine	$19.50	2393	Brain Game	$19.50
2381	Outlaw	$19.50	2394	Superman	$19.50
2382	Slot Racer	$19.50	2395	Casino	$19.50
2383	Video Olympics	$19.50	2396	Backgammon	$37.95
2384	Breakout	$19.50	2397	Video Chess	$37.95
2385	Canyon Bomber	$19.50	2500	Space Invaders	$25.00
2386	Street Racer	$19.50	2609	Adventure	$25.00
2387	Homerun	$19.50	2611	Indy 500	$37.95
2388	Basketball	$19.50		Game Cartridge Weight: 6 oz.	

Leedex VIDEO 100 12" MONITOR

- *Compatible with TRS-80 (no interface required)*
- *Compatible with many home computers!*

Now from LEEDEX . . . One of the most popular low cost, yet high resolution (650 line) monitors currently available. These units compare favorably with monitors costing twice as much. Because of the fact that standard composite video input is utilized no RF modulator is needed. An extremely sharp and stable picture is achieved. The video bandwidth is 12 mhz +/− 3db with a 75 ohm input impedance.

Cat No. 1204	Video 100 Monitor	Wt. 18 lb.	$139.00
Cat No. 1937	TRS-80/Leedex cable kit	Wt. 6 oz.	$3.00

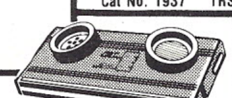
16K MEMORY ADD-ON KIT $60.00

Everything you need to upgrade your TRS-80, Apple or Exidy! An additional 16K includes illustrated instructions, RAMS and preprogrammed jumpers. No special tools required. Wt. 4 oz.

CAT NO	DESCRIPTION
1156	TRS-80 Keyboard Unit
1156-A	TRS-80 Exp. Interface (prior to 4/1/79)
1156-B	TRS-80 Exp. Interface (after 4/1/79)
1156-C	for Apple II
1156-D	for Exidy

CCS 7811B ARITHMETIC PROCESSOR

Assembled & tested, adds advanced arithmetic power to your Apple II. AMD AM 9511 based, 16 and 32 bit fixed point, 32 bit floating point operation. Float to fixed and fixed to float conversions. Trig and inverse functions, square roots, logs, exponentiation. Interrupt daisy chain, DMA daisy chain, and much more. Wt. 2 lbs.
Cat No. 1635 **$399.95**



NOVATION "CAT" ACOUSTIC MODEM $179.00

The first compact modem designed for the small computer user. Transmits data over standard telephone lines. Exchange data or program with other systems. Data transfer rate up to 30 char/sec. Complete and ready to use! Requires 110 VAC. 60 Hz.
Cat No. 1480 Wt. 3 lbs.

SEND FOR FREE FLYER, FEATURING:

Page after page of exciting products. Computerized toys and games, personal computers, disk drives, integrated circuits, semi conductors. Add new dimension to your Apple, Atari, TRS-80, etc. with our special application boards and comprehensive software library. Hundreds of products available at terrific Hobbyworld prices. Circle our reader service number or write/phone for your free illustrated flyer today.

HOW TO ORDER

Pay by check, Mastercharge, Visa, or C.O.D. Charge card orders please include expiration date. Payment in U.S. dollars only. Order by phone, mail or at our retail store. MINIMUM ORDER $10.00. Please include phone number and magazine issue you are ordering from. Prices valid thru last day of cover date. SHIPPING: USA: Add $2.00 for first 2 lbs., 35¢ each addt'l lb. for ground. For AIR add $3.00 first 2 lbs., 75¢ each addt'l lb. FOREIGN: surface: $3.00 first 2 lbs., 60¢ each addt'l lb. AIR: $11.00 first 2 lbs., $5.00 each addt'l lb. CODs: add $1.25 addt'l. Not responsible for typographical errors. Some items subject to prior sale or quantity limitations. 120 day guaranteed satisfaction. Exception: partially assembled kits.

VERBATIM 5¼" DISKETTES

10 per box

CAT NO.	TYPE	DESCRIPTION	PRICE
1147	525-01	Soft sector, TRS-80, etc.	$33.00
1148	525-10	10 hole, hard, Apple, North Star	$33.00
1149	525-16	16 hole, hard, Micropolis	$33.00
2330	577-01	Soft sector, certified	$49.95
2331	577-10	10 hole, hard, certified	$49.95
2332	577-16	16 hole, hard, certified	$49.95

DISKETTE DRIVE HEAD CLEANING KIT

Diskette drive heads need periodic maintenance to assure efficient and error-free operation. Unlike other peripheral devices, the read/write head(s) on diskette drives are extremely difficult to clean without partially disassembling the drive. The diskette drive head cleaning kit allows the user to clean the heads in just minutes, without disassembling the drive. Available in 5½" or 8", both single and double sided. Kit contains 2 cleaning diskettes, 4 oz. bottle of CS-85 cleaning solution and an easy pour dispenser.
Weight 12 oz.

CAT NO.	DESCRIPTION	PRICE
2499	8" Disk Drive Cleaning Kit	$30.75
2534	5¼" Disk Drive Cleaning Kit	$30.75

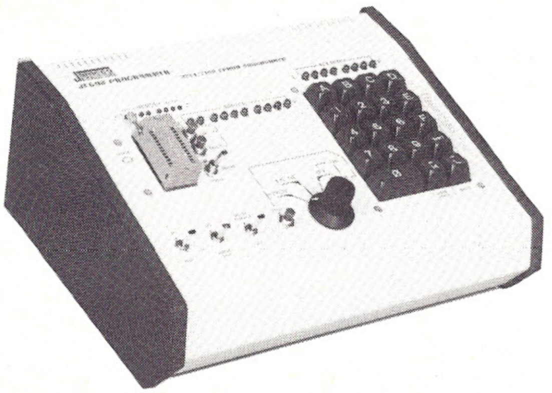
19511 Business Center Drive Dept. K7 Northridge, Calif. 91324

NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW! NEW!

Jameco ELECTRONICS KITS

JE608 PROGRAMMER

2704/2708 EPROM PROGRAMMER



- 3 separate Display Registers: 8 LED's for Hex Key entries, 10 LED's (2^0-2^9) for Address Register and 8 LED's for Data Memory Register. The Data Memory Register displays the content of the EPROM.
- Development of microprocessor systems by means of a ribbon cable from the programmer panel test socket to the EPROM socket on the microprocessor board.
- Rapid checking verification of programmed data changes.
- User may move data from a master to RAM's or write into RAM's with keyboard entries.
- Allows manual stepping manipulation (up and down) at any address location.
- Stand-alone EPROM Programmer consisting of: A 19-key Hexadecimal Keyboard assembly, Programmer Board assembly with 4 power supplies and a LED/Test Socket Panel Board assembly. The Test Socket is zero force insertion type. Power requirements: 115VAC, 60Hz, 6W.
- Compact desk-top enclosure: Color-coordinated designer's case with light tan panels and end pieces in molded mocha brown. Size: 3¼"H x 11"W x 8¾"D. Weight: 5 lbs.

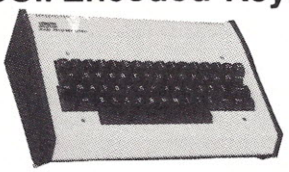
GENERAL APPLICATIONS:

- TO PROGRAM EPROMS 2704 and 2708.
- DEVELOPMENTAL SYSTEM FOR MICRO-COMPUTER CIRCUITS
- TO READ THE CONTENTS OF A PRE-PROGRAMMED EPROM.
- TO COMPARE EPROM(S) FOR CONTENT DIFFERENCES
- TO EMULATE A PROGRAMMED EPROM
- TO STORE PROGRAM IN RAMS FOR ALTERATIONS

The JE608 EPROM Programmer is a completely self-contained unit which is independent of computer control and requires no additional systems for its operations. The EPROM can be programmed from the Hexadecimal Keyboard or from a pre-programmed EPROM. The JE608 Programmer can emulate a programmed EPROM by the use of its internal RAM circuits. This will allow the user to test or pretest a program, for a system, prior to programming a chip. Any changes in the program can be entered directly into the memory circuits with the Hexadecimal Keyboard so that rewriting the entire program will not be necessary. The JE608 Programmer contains a Programmer/Board with 25 IC's and including power supplies of: −5V, +5V, +12V and +26V. The Hexadecimal Keyboard and LED/ Test Socket Panel board are separate assemblies within the system.

JE608 Kit $399.95
Assembled and tested . . $499.95

JE610 ASCII Encoded Keyboard Kit



The **JE610 ASCII Keyboard Kit** can be interfaced into most any computer system. The kit comes complete with an industrial grade keyboard switch assembly (62-keys), IC's, sockets, connector, electronic components and a double-sided printed wiring board. The keyboard assembly requires +5V @ 150mA and −12V @ 10mA for operation. Features: 60 keys generate the full 128 characters, upper and lower case ASCII set. Fully buffered. Two user-define keys provided for custom applications. Caps lock for upper-case-only alpha characters. Utilizes a 2376 (40-pin) encoder read-only memory chip. Outputs directly compatible with TTL/DTL or MOS logic arrays. Easy interfacing with a 16-pin dip or 18-pin edge connector.

JE610 ASCII Encoded Keyboard Kit only . . $79.95

Desk-Top Enclosure for JE610 ASCII Encoded Keyboard Kit

Compact desk-top enclosure: Color-coordinated designer's case with light tan aluminum panels and molded end pieces in mocha brown. Includes mounting hardware. Size: 3½"H x 14½"W x 8¾"D.

DTE-AK $49.95

SPECIAL: JE610/DTE-AK PURCHASED TOGETHER (Value $129.90) $124.95

JE600 Hexadecimal Encoder Kit

FULL 8-BIT
LATCHED OUTPUT
19-KEY KEYBOARD

The **JE600 Encoder Keyboard Kit** provides two separate hexadecimal digits produced from sequential key entries to allow direct programming for 8-bit microprocessor or 8-bit memory circuits. Three additional keys are provided for user operations with one having a bistable output available. The outputs are latched and monitored with 9 LED readouts. Also included is a key entry strobe. Features: Full 8-bit latched output for microprocessor use. Three user-define keys with one being bistable operation. Debounce circuit provided for all 19 keys. 9 LED readouts to verify entries. Easy interfacing with standard 16-pin IC connector. Only +5VDC required for operation.

JE600 Hexadecimal Board Kit only $59.95

Desk-Top Enclosure for JE600 Hexadecimal Keyboard Kit

Compact desk-top enclosure: Color-coordinated designer's case with light tan aluminum panels and molded end pieces in mocha brown. Includes mounting hardware. Size: 3½"H x 8¼"W x 8¾"D.

DTE-HK $44.95

SPECIAL: JE600/DTE-HK PURCHASED TOGETHER (Value $104.90) $99.95

AVAILABLE TODAY ONLY THROUGH JAMECO!

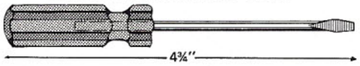
7-80

Jameco® ELECTRONICS

$10.00 MINIMUM ORDER — U.S. FUNDS ONLY — CALIF. RESIDENTS ADD 6% SALES TAX — POSTAGE: ADD 5% PLUS $1.00 INSURANCE — 1980 CATALOG AVAILABLE (IF DESIRED SEND 41-CENT STAMP TO:

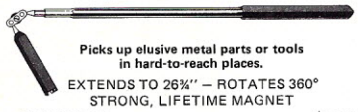
1355 Shoreway Road • Belmont, CA 94002 • Phone orders welcome (415) 592-8097

MINIATURE SCREWDRIVER



RUGGED CONSTRUCTION
1000's OF USES – 1/8" TIP . . . . **2/$1.00**

MAGNETIC RETRIEVER TOOL



Picks up elusive metal parts or tools in hard-to-reach places.

EXTENDS TO 26¾" – ROTATES 360°
STRONG, LIFETIME MAGNET

MRT-2281 . . . . $3.95

CSC CONTINENTAL SPECIALTIES

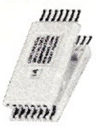
LOGIC PROBE KIT

SPECIFICATIONS

Input Impedance: 300,000 Ohms.
Thresholds: "Lo" 30%Vcc – "Hi" 70%Vcc
Maximum Speed: 300 nsec., 1.5MHz
Input Protection: ±50VDC continuous 117VAC for 15 sec.
Power: 30mA @5V – 40mA @ 15V – 25V max. reverse voltage protected; 36" cable with color coded clips included.
Operating Temp.: 0-50°C.
Dimensions: 5.8L x 1.0W x 0.7D in. (147 x 25 x 18mm)
Weight: 30 oz. (85 gm)

LPK-1 . . . . $21.95/Kit

Proto Clips

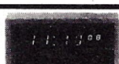


14-PIN CLIP	PC-14	$ 4.50
16-PIN CLIP	PC-16	$ 4.75
24-PIN CLIP	PC-24	$10.00
40-PIN CLIP	PC-40	$16.00

Proto Boards

PB-6	$17.95
PB-100	19.95
PB-101	22.95
PB-102	26.95
PB-103	44.95
PB-104	55.95
PB-203	99.95
PB-203A	155.00
PB-203A-Kit	131.00

Jumbo 6-Digit Clock Kit



- Four .630"ht. and two .300"ht. common anode displays
- Uses MM5314 clock chip
- Switches for hours, minutes and hold functions
- Hours easily viewable to 30 feet
- Simulated walnut case
- 115VAC operation
- 12 or 24 hour operation
- Includes all components, case and wall transformer
- Size: 6¾ x 3⅛ x 1¾

JE747 . . . . $29.95

- Bright .300 ht. comm. cathode display
- Uses MM5314 clock chip
- Switches for hours, minutes and hold modes
- Hrs. easily viewable to 20 ft.
- Simulated walnut case
- 115 VAC operation
- 12 or 24 hr. operation
- Incl. all components, case & wall transformer
- Size: 6¾" x 3-1/8" x 1¾"

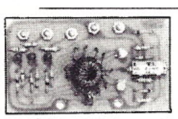
JE701

6-Digit Clock Kit $19.95

Regulated Power Supply

Uses LM309K. Heat sink provided. PC board construction. Provides a solid 1 amp @ 5 volts. Can supply up to ±5V, ±9V and ±12V with JE205 Adapter. Includes components, hardware and instructions. Size: 3½" x 5" x 2"H

JE200 . . . . $14.95



ADAPTER BOARD —Adapts to JE200— ±5V, ±9V and ±12V

DC/DC converter with +5V input. Toriodal high-speed switching XMFR. Short circuit protection. PC board construction. Piggy-back to JE 200 board. Size: 3½" x 2" x 9/16"H

JE205 . . . . $12.95

MICROPROCESSOR COMPONENTS

8080A/8080A SUPPORT DEVICES

8080A	CPU	$ 7.95
8212	8-Bit Input/Output	3.25
8214	Priority Interrupt Control	5.95
8216	Bi-Directional Bus Driver	3.49
8224	Clock Generator/Driver	3.95
8226	Bus Driver	3.49
8228	System Controller/Bus Driver	4.95
8238	System Controller	5.95
8251	Prog. Comm. 1/0 (USART)	7.95
8253	Prog. Interval Timer	14.95
8255	Prog. Periph. 1/0 (PPI)	9.95
8257	Prog. DMA Control	19.95
8259	Prog. Interrupt Control	19.95

6800/6800 SUPPORT DEVICES

MC6800	MPU	$14.95
MC6802CP	MPU with Clock and Ram	24.95
MC6810API	128X8 Static Ram	5.95
MC6821	Periph. Inter. Adapt (MC6820)	7.49
MC6828	Priority Interrupt Controller	12.95
MC6830L8	1024X8 Bit ROM (MC68A30-8)	14.95
MC6850	Asynchronous Comm. Adapter	7.95
MC6852	Synchronous Serial Data Adapt.	9.95
MC6860	0-600 bps Digital MODEM	12.95
MC6862	2400 bps Modulator	14.95
MC6880A	Quad 3-State Bus. Trans. (MC8T26)	2.25

MICROPROCESSOR CHIPS—MISCELLANEOUS

Z80(780C)	CPU	$13.95
Z80A(780-1)	CPU	15.95
CDP1802	CPU	19.95
2650	MPU	19.95
6502	CPU	11.95
8035	8-Bit MPU w/clock, RAM, 1/0 lines	19.95
P8085	CPU	19.95
TMS9900JL	16-Bit MPU w/hardware, multiply & divide	49.95

SHIFT REGISTERS

MM500H	Dual 25 Bit Dynamic	$.50
MM503H	Dual 50 Bit Dynamic	.50
MM504H	Dual 16 Bit Static	.50
MM506H	Dual 100 Bit Static	.50
MM510H	Dual 64 Bit Accumulator	.50
MM5016H	500/512 Bit Dynamic	.89
2504T	1024 Dynamic	3.95
2518	Hex 32 Bit Static	4.95
2522	Dual 132 Bit Static	2.95
2524	512 Static	.99
2525	1024 Dynamic	2.95
2527	Dual 256 Bit Static	2.95
2528	Dual 250 Static	4.00
2529	Dual 240 Bit Static	4.00
2532	Quad 80 Bit Static	2.95
3341	Fifo	6.95
74LS670	4X4 Register File (TriState)	2.49

UART'S

A-Y-5-1013	30K BAUD	5.95

MICROPROCESSOR MANUALS

M-Z80	User Manual	$7.50
M-CDP1802	User Manual	7.50
M-2650	User Manual	5.00

ROM'S

2513(2140)	Character Generator(upper case)	$9.95
2513(3021)	Character Generator(lower case)	9.95
2516	Character Generator	10.95
MM5230N	2048-Bit Read Only Memory	1.95

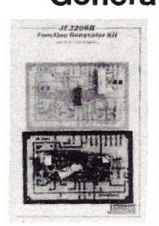
RAM'S

1101	256X1	Static	$1.49
1103	1024X1	Dynamic	.99
2101(8101)	256X4	Static	3.95
2102	1024X1	Static	1.75
21L02	1024X1	Static	1.95
2111(8111)	256X4	Static	3.95
2112	256X4	Static MOS	4.95
2114	1024X4	Static 450ns	7.95
2114L	1024X4	Static 450ns low power	10.95
2114-3	1024X4	Static 300ns	10.95
2114L-3	1024X4	Static 300ns low power	11.95
5101	256X4	Static	7.95
5280/2107	4096X1	Dynamic	4.95
7489	16X4	Static	1.75
74S200	256X1	Static Tristate	4.95
93421	256X1	Static	2.95
UPD414 (MK4027)	4K	Dynamic 16 pin	4.95
UPD416 (MK4116)	16K	Dynamic 16 pin 250ns	7.95
TMS4044-45NL	4K	Static	14.95
TMS4045	1024X4	Static	14.95
2117	16,384X1	Dynamic 350ns (house marked)	9.95
MM5262	2KX1	Dynamic	4/1.00

PROM'S

1702A	2048	FAMOS	$5.95
2716INTEL	16K*	EPROM	59.95
TMS2516 (2716)	16K*	EPROM	24.95
	*Requires single +5V power supply		
TMS2532	4KX8	EPROM	89.95
2708	8K	EPROM	10.95
2716 T.I	16K**	EPROM	29.95
	**Requires 3 voltages, —5V, +5V, +12V		
5203	2048	FAMOS	14.95
6301-1(7611)	1024	Tristate Bipolar	3.49
6330-1(7602)	256	Open C Bipolar	2.95
82S23	32X8	Open Collector	3.95
82S115	4096	Bipolar	19.95
82S123	32X8	Tristate	3.95
74186	512	TTL Open Collector	9.95
74188	256	TTL Open Collector	3.95
74S287	1024	Static	2.95

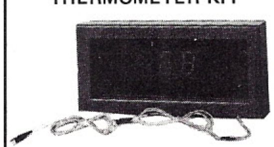
Function Generator Kit



Provides 3 basic waveforms: sine, triangle and square wave. Freq. range from 1 Hz to 100K Hz. Output amplitude from 0 volts to over 6 volts (peak to peak). Uses a 12V supply or a ±6V split supply. Includes chip, P.C. Board, components & instructions.

JE2206B $19.95

DIGITAL THERMOMETER KIT



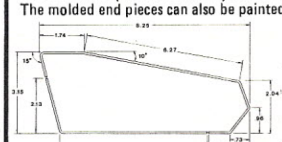
- Dual sensors—switching control for indoor/outdoor or dual monitoring
- Continuous LED .8" ht. display
- Range: -40°F to 199°F / -40°C to 100°C
- Accuracy: ±1° nominal
- Set for Fahrenheit or Celsius reading
- Sim. walnut case - AC wall adapter incl.
- Size: 3-1/4"H x 6-5/8"W x 1-3/8"D

JE300 . . . . $39.95

DESIGNERS' SERIES
Blank Desk-Top Electronic Enclosures

- **High strength epoxy molded end pieces in mocha brown finish.**
- **Sliding rear/bottom panel for service and component accessibility.**
- **Top / bottom panels .080 thk alum. Alodine type 1200 finish (gold tint color) for best paint adhesion after modification.**
- **Vented top and bottom panels for cooling efficiency.**
- **Rigid construction provides unlimited applications.**

CONSTRUCTION:
The "DTE" Blank Desk Top Electronic Enclosures are designed to blend and complement today's modern computer equipment and can be used in both industrial and home. The end pieces are precision molded with an internal slot (all around) to accept both top and bottom panels. The panels are then fastened to ¼" thick tabs inside the end pieces to provide maximum rigidity to the enclosure. For ease of equipment servicing, the rear/bottom panel slides back on slotted tracks while the rest of the enclosure remains intact. Different panel widths may be used while maintaining a common profile outline. The molded end pieces can also be painted to match any panel color scheme.



Enclosure Model No.	Panel Width	PRICE
DTE-8	8.00"	$29.95
DTE-11	10.65"	$32.95
DTE-14	14.00"	$34.95

$10.00 Min. Order – U.S. Funds Only
Calif. Residents Add 6% Sales Tax
Postage—Add 5% plus $1 Insurance (if desired)

Spec Sheets – 25¢
1980 Catalog Available – Send 41¢ stamp

FREE 1980 CATALOG

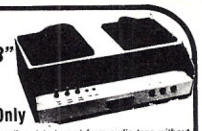
Jameco ELECTRONICS

✓41 **PHONE ORDERS WELCOME (415) 592-8097**

MAIL ORDER ELECTRONICS – WORLDWIDE
1355 SHOREWAY ROAD, BELMONT, CA 94002
PRICES SUBJECT TO CHANGE

7-80

The Incredible "Pennywhistle 103"



$139.95 Kit Only

The Pennywhistle 103 is capable of recording data to and from audio tape without critical speed requirements for the recorder and it is able to communicate directly with another modern and terminal for telephone "hamming" and communications. In addition, it is free of critical adjustments and is built with non-precision, readily available parts.

Data Transmission Method . . . . . Frequency-Shift Keying, full-duplex (half-duplex selectable).
Maximum Data Rate . . . . 300 Baud.
Data Format . . . . Asynchronous Serial (return to mark level required between each character).
Receive Channel Frequencies . . . 2025 Hz for space; 2225 Hz for mark.
Transmit Channel Frequencies . . Switch selectable: Low (normal) = 1070 space, 1270 mark; High = 025 space, 2225 mark.
Receive Sensitivity . . . . −46 dbm accoustically coupled.
Transmit Level . . . . −15 dbm nominal. Adjustable from −6 dbm to −20 dbm.
Receive Frequency Tolerance . . . Frequency reference automatically adjusts to allow for operation between 1800 Hz and 2400 Hz.
Digital Data Interface . . . . EIA RS-232C or 20 mA current loop (receiver is optoisolated and non-polar).
Power Requirements . . . . 120 VAC, single phase, 10 Watts.
Physical . . . . All components mount on a single 5" by 9" printed circuit board. All components included.

Requires a VOM, Audio Oscillator, Frequency Counter and/or Oscilloscope to align.

TRS-80 16K Conversion Kit

Expand your 4K TRS-80 System to 16K.
Kit comes complete with:
- **8 each UPD416-1 (16K Dynamic Rams) 250NS**
- **Documentation for conversion**

TRS-16K $59.95

ok JUST WRAP™

- **30 AWG wire**
- **Daisy chain or point-to-point**
- **No stripping or slitting required—just wrap**
- **.025" square posts**
- **Built-in cut off**
- **Includes 50 ft. wire**

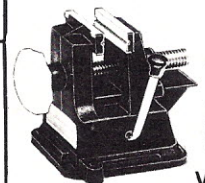
Part No.	Wire Color	Price
JW-1-B	Blue	$14.95
JW-1-W	White	14.95
JW-1-Y	Yellow	14.95
JW-1-R	Red	14.95

JUST WRAP™ Replacement Wire

Part No.	Color		Price
R-JW-B	Blue	50 ft. roll	$2.98
R-JW-W	White	50 ft. roll	2.98
R-JW-Y	Yellow	50 ft. roll	2.98
R-JW-R	Red	50 ft. roll	2.98

JUST WRAP™ Unwrap Tool $3.49

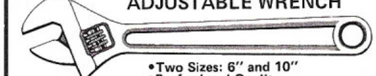
ok Vacuum Vise



Vacuum-based light-duty vise for small components and assemblies. ABS construction. 1½" jaws, 1¼" travel. Can be permanently installed.

VV-1 . . . . . . $3.49

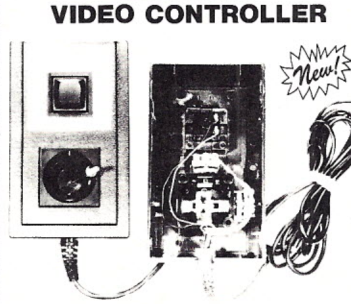
ADJUSTABLE WRENCH



- **Two Sizes: 6" and 10"**
- **Professional Quality**
- **Chrome Vanadium Steel**

6"— AW-6 . . $4.95 10"— AW-10 . . $6.95

JOYSTICK VIDEO CONTROLLER



- IDEAL FOR ALL VIDEO GAMES OR REMOTE CONTROL PROJECTS
- SMALL CASE SIZE: 1-1/2"H x 2-3/8"W x 4-5/16"L
- 2 MINIATURE POTENTIOMETERS—40K OHM EACH
- SPST PUSH BUTTON CONTROL
- 5-WIRE CONNECTION CABLE – 5 FEET LONG
- RUGGED PLASTIC CASE

JVC-40 . . . . . . $4.95 each

DIGITAL RESEARCH COMPUTERS

(214) 271-3538

32K S-100 EPROM CARD

NEW!

$74.95 KIT

USES 2716's

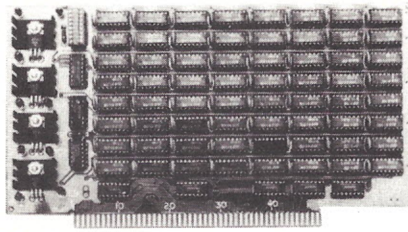
Blank PC Board - $34

ASSEMBLED & TESTED ADD $30

SPECIAL: 2716 EPROM's (450 NS) Are $19.95 EA. With Above Kit

KIT FEATURES:
1. Uses +5V only 2716 (2Kx8) EPROM's.
2. Allows up to 32K of software on line!
3. IEEE S-100 Compatible.
4. Addressable as two independent 16K blocks.
5. Cromemco extended or Northstar bank select.
6. On board wait state circuitry if needed.
7. Any or all EPROM locations can be disabled.
8. Double sided PC board, solder-masked, silk-screened.
9. Gold plated contact fingers.
10. Unselected EPROM's automatically powered down for low power.
11. Fully buffered and bypassed.
12. Easy and quick to assemble.

8K LOW POWER RAM KIT-S 100 BUSS



21L02 (450 NS RAMS!)

$119.50 KIT

ASSEMBLED & FULLY BURNED IN ADD $35

Thousands of computer systems rely on this rugged, work horse, RAM board. Designed for error-free, NO HASSLE, systems use

ADD $10 FOR 4 MHZ

Blank PC Board w/Documentation - **$29.95**
Low Profile Socket Set - **$13.50**
Support IC's (TTL & Regulators) - **$9.75**
Bypass CAP's (Disc & Tantalums) - **$4.50**

FOR 4MHZ SALE! **LOW POWER - 300NS** **2114 RAM SALE!** **8 FOR $44**

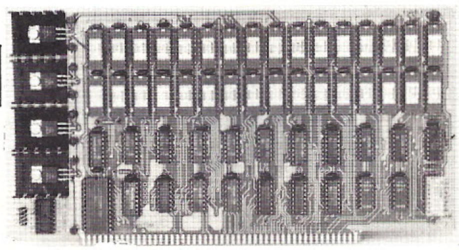
4K STATIC RAM'S. MAJOR BRAND, NEW PARTS.
These are the most sought after 2114's, LOW POWER and 300NS FAST.
8 FOR $44

16K STATIC RAM KIT-S 100 BUSS

PRICE CUT!

$225 KIT

FOR 4MHZ ADD $10



KIT FEATURES:
1. Addressable as four separate 4K Blocks.
2. ON BOARD BANK SELECT circuitry. (Cromemco Standard!). Allows up to 512K on line!
3. Uses 2114 (450NS) 4K Static Rams.
4. ON BOARD SELECTABLE WAIT STATES.
5. Double sided PC Board, with solder mask and silk screened layout. Gold plated contact fingers.
6. All address and data lines fully buffered.
7. Kit includes ALL parts and sockets.
8. PHANTOM is jumpered to PIN 67.
9. LOW POWER: under 1.5 amps TYPICAL from the +8 Volt Buss.
10. Blank PC Board can be populated as any multiple of 4K.

BLANK PC BOARD W/DATA-**$33**
LOW PROFILE SOCKET SET-**$12**
SUPPORT IC'S & CAPS-**$19.95**
ASSEMBLED & TESTED-**ADD $35**

OUR #1 SELLING RAM BOARD!

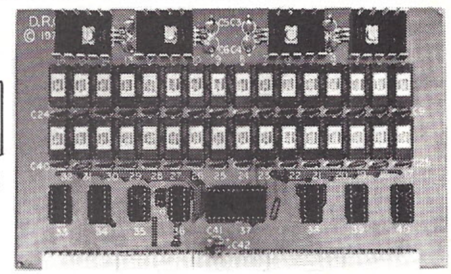
16K STATIC RAM SS-50 BUSS

PRICE CUT!

$229 KIT

FULLY STATIC!

FOR 2MHZ ADD $10



FOR SWTPC 6800 BUSS!

ASSEMBLED AND TESTED - $35

KIT FEATURES:
1. Addressable on 16K Boundaries
2. Uses 2114 Static Ram
3. Fully Bypassed
4. Double sided PC Board. Solder mask and silk screened layout.
5. All Parts and Sockets included
6. Low Power: Under 1.5 Amps Typical

BLANK PC BOARD—**$26** COMPLETE SOCKET SET—**$12**
SUPPORT IC'S AND CAPS—**$19.95**

NEW! STEREO! NEW! S-100 SOUND COMPUTER BOARD

At last, an S-100 Board that unleashes the full power of two unbelievable General Instruments AY3-8910 NMOS computer sound IC's. Allows you under total computer control to generate an infinite number of special sound effects for games or any other program. Sounds can be called in BASIC, ASSEMBLY LANGUAGE, etc.

COMPLETE KIT! $84.95 (WITH DATA MANUAL)

BLANK PC BOARD W/DATA $31

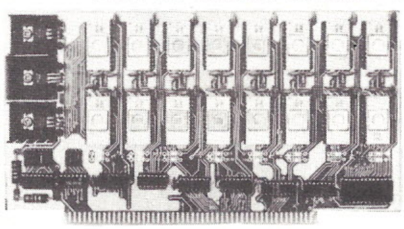
KIT FEATURES:
- ★ TWO GI SOUND COMPUTER IC'S.
- ★ FOUR PARALLEL I/O PORTS ON BOARD.
- ★ USES ON BOARD AUDIO AMPS OR YOUR STEREO.
- ★ ON BOARD PROTO TYPING AREA.
- ★ ALL SOCKETS, PARTS AND HARDWARE ARE INCLUDED.
- ★ PC BOARD IS SOLDERMASKED, SILK SCREENED, WITH GOLD CONTACTS.
- ★ EASY, QUICK, AND FUN TO BUILD. WITH FULL INSTRUCTIONS.
- ★ USES PROGRAMMED I/O FOR MAXIMUM SYSTEM FLEXIBILITY.

Both Basic and Assembly Language Programming examples are included.

SOFTWARE:

SCL™ is now available! Our Sound Command Language makes writing Sound Effects programs a SNAP! SCL™ also includes routines for Register-Examine-Modify, Memory-Examine-Modify, and Play-Memory. SCL™ is available on CP/M* compatible diskette of 2708 or 2716. Diskette - **$24.95** 2708 - **$19.95** 2716 - **$29.95** Diskette includes the source. EPROM'S are ORG at E000H.

16K EPROM CARD-S 100 BUSS



$59.95 KIT

BLANK PC BOARD - $28

USES 2708's!

Thousands of personal and business systems around the world use this board with complete satisfaction. Puts 16K of software on line at **ALL TIMES!** Kit features a top quality soldermasked and silk-screened PC board and first run parts and sockets. Any number of EPROM locations may be disabled to avoid any memory conflicts. Fully buffered and has WAIT STATE capabilities.

ASSEMBLED AND FULLY TESTED — ADD $30

OUR 450 NS 2708'S ARE $8.95 EA. WITH PURCHASE OF KIT

RCA CMOS COMPUTER CHIP SET

INCLUDES:

1-CDP1802CD CPU	1-CDP1861CD VIDEO IC
2-CDP1822CE 256 x 4 RAM	1-CDP1862CE COLOR GEN.
1-CDP1858CE 4 BIT LATCH	1-CDP1863CE SOUND GEN.

COMPLETE SET $45 LIMITED QTY

NEW! G.I. COMPUTER SOUND CHIP

AY3-8910. As featured in July, 1979 BYTE! A fantastically powerful Sound & Music Generator. Perfect for use with any 8 Bit Microprocessor. Contains: 3 Tone Channels, Noise Generator, 3 Channels of Amplitude Control. 16 bit Envelope Period Control, 2-8 Bit Parallel I/O. 3 D to A Converters, plus much more! All in one 40 Pin DIP. Super easy interface to the S-100 or other busses.

SPECIAL OFFER: $14.95 each Add $3 for 60 page Data Manual.

Digital Research Computers
(OF TEXAS)

P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Add $1.00 postage. we pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. 90 Day Money Back Guarantee on all items. Orders over $50. add 85¢ for insurance.

***TRADEMARK OF DIGITAL RESEARCH. NOT ASSOCIATED WITH DIGITAL RESEARCH OF CALIFORNIA, THE SUPPLIERS OF CPM SOFTWARE.**

WAMECO

THE COMPLETE PC BOARD HOUSE
EVERYTHING FOR THE S-100 BUSS

INTRODUCTORY SPECIAL
IOB-1 SERIAL/PARALLEL INTERFACE BOARD

* TWO PARALLEL DATA PORTS PROGRAMMABLE USING AN 8255 WITH SEPARATE HANDSHAKING.
* ONE SERIAL PORT USING AN 8251 WITH PROVISIONS FOR PARITY, STOP BIT AND CHARACTER LENGTH. BAUD RATES 110 TO 9600 BAUD. OUTPUTS RS232, TTL AND CURRENT LOOP.
* KANSAS CITY STANDARD CASSETTE INTERFACE, 300 BAUD FOR USE WITH THE SERIAL INTERFACE.
* STATUS MAY BE POLLING SOFTWARE OR VECTURED INTERRUPTS.

PCBD $31.95

KIT TO BE ANNOUNCED LATER.

FUTURE PRODUCTS: 80 CHARACTER VIDEO BOARD.

Z-80 CPU BOARD WITH ROM, 8 PARALLEL PORT I/O BOARD.

DEALER INQUIRIES INVITED, UNIVERSITY DISCOUNTS AVAILABLE AT YOUR LOCAL DEALER

MOST PRODUCTS FOR IMMEDIATE SHIPMENT. NO 4-8 WEEK DELAYS REQUIRED FOR OTHERS.

NOTE: NEW ADDRESS AND PHONE

WMC inc.

WAMECO, INC., P. O. BOX 877 • 455 PLAZA ALHAMBRA • EL GRANADA, CA 94018 • (415) 726-6378

CALIFORNIA COMPUTER SYSTEMS

16K RAM BOARD. Fully buffered addressable in 4K blocks. IEEE standard for bank addressing 2114's. PCBD .... $27.95 Kit 450 NSEC $249.95

PT-1 PROTO BOARD. Over 2,600 holes 4" regulators. All S-100 buss functions labeled, gold fingers. PCBD $26.95

PT-2 PROTO BOARD. Similar to PT-1 except set-up to handle solder tail sockets. PCBD ... $26.95

CCS MAIN FRAME. Kit (S-100) $339.95

APPLE EXTENDER. Kit $22.95

APPLE IEEE INSTRUMENTATION INTERFACE KIT 7490. Kit $275.00

ARITHMETIC PROCESSOR FOR APPLE 7811A. Kit $350.00

APPLE ASYNCHRONOUS SERIAL INTERFACE 7710A. Kit $89.95

APPLE SYNCHRONOUS SERIAL INTERFACE 7712A. Kit $89.95

ALL OTHER CCS PRODUCTS AVAILABLE

SSM

PB-1 2708 & 2716 Programming Board with provisions for 4K or 8K EPROM. No external supplies required. Textool sockets. Kit $129.95

CB-1A 8080 Processor Board. 2K of PROM 256 BYTE RAM power on/rest Vector Jump Parallel port with status. Kit .... $129.95 PCBD $27.95

VB-3 80 x 55 VIDEO BOARD. Graphic included. 4 MHZ **$379.95**

IO-4 Two serial I/O ports with full handshaking 20/60 ma current loop: Two parallel I/O ports. Kit .... $130.00 PCBD $27.95

VB-1B 64 x 16 video board, upper lower case Greek composite and parallel video with software, S-100. Kit .... $125.00 PCBD $27.95

CB-2 Z80 CPU BOARD. Kit $185.95

AIO APPLE SERIAL/PARALLEL $125.95

ALL OTHER SSM PRODUCTS AVAILABLE

WMC inc. WAMECO INC.

FDC-1 FLOPPY CONTROLLER BOARD will drive shugart, pertek, remic 5" & 8" drives up to 8 drives, on board PROM with power boot up, will operate with CPM™ (not included). PCBD $42.95

FPB-1 Front Panel. IMSAI size, hex displays. Byte, or instruction single step. PCBD $47.50

MEM-1A 8K x 8 fully buffered, S-100, uses 2102 type rams. PCBD $25.95

QM-12 MOTHER BOARD, 13 slot, terminated, S-100 board only $38.95

CPU-1 8080A Processor board S-100 with 8 level vector interrupt. PCBD $27.95

RTC-1 Realtime clock board. Two independent interrupts. Software programmable. PCBD $24.95

EPM-1 1702A 4K Eprom card. PCBD $25.95

EPM-2 2708/2716 16K/32K EPROM CARD. PCBD $25.95

QM-9 MOTHER BOARD. Short Version of QM-12. 9 Slots. PCBD $32.95

MEM-2 16K x 8 Fully Buffered 2114 Board. PCBD $27.95

PTB-1 POWER SUPPLY AND TERMINATOR BOARD. PCBD $27.95

IOB-1 SERIAL AND PARALLEL INTERFACE. 2 parallel, one serial and cassette. PCBD $27.95

2708	$ 9.49	2114L 450 NSEC	$5.99
2716	$35.95	2114L 200 NSEC	$6.99

JULY SPECIAL SALE ON PREPAID ORDERS
(Charge cards not included on this offer)

FPB-1 with MIKOS #14 all parts for front panel $129.95

MIKOS PARTS ASSORTMENT WITH WAMECO AND CYBERCOM PCBDS

MEM-2 with MIKOS #7 16K ram with L2114 450 NSEC $249.95

MEM-2 with MIKOS #13 16K ram with L2114 250 NSEC $279.95

CPU-1 with MIKOS #2 8080A CPU .... $ 94.95

QM-12 with MIKOS #4 13 slot mother board .... $ 95.95

RTC-1 with MIKOS #5 real time clock .... $ 59.95

EMP-1 with MIKOS #10 4K 1702 less EPROMS .... $ 49.95

EPM-2 with MIKOS #11 16-32K EPROMS less EPROMS .... $ 59.95

QM-9 with MIKOS #12 9 slot mother board .... $ 89.95

FPB-1 with MIKOS #14 all parts for front panel $144.95

MIKOS PARTS ASSORTMENTS ARE ALL FACTORY MARKED PARTS. KITS INCLUDE ALL PARTS LISTED AS REQUIRED FOR THE COMPLETE KIT LESS PARTS LISTED. ALL SOCKETS INCLUDED.

MIKOS

NOTE: NEW ADDRESS AND PHONE

(415) 726-7593

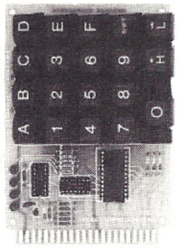
P. O. Box 955 • El Granada, CA 94018

Please send for IC, Xistor and Computer parts list

VISA or MASTERCHARGE. Send account number, interbank number, expiration date and sign your order. Approx. postage will be added. Check or money order will be sent post paid in U.S. If you are not a regular customer, please use charge, cashier's check or postal money order. Otherwise there will be a two-week delay for checks to clear. Calif. residents add 6% tax. Money back 30-day guarantee. We cannot accept returned IC's that have been soldered to. Prices subject to change without notice. **$10 minimum order. $1.50 service charge on orders less than $10.00.**

HEX ENCODED KEYBOARD

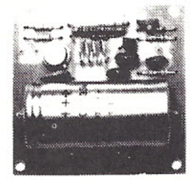
Four onboard LEDs indicate the HEX code generated for each key depression. The board requires a single +5 volt supply. Board only $15.00 Part No. HEX-3, with parts $49.95 Part No. HEX- 3A. 44 pin edge connector $4.00 Part No. 44P.



ASCII TO CORRESPONDENCE CODE CONVERTER

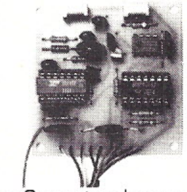
This bidirectional board is a direct replacement for the board inside the Trendata 1000 terminal. The on board connector provides RS-232 serial in and out. Sold only as an assembled and tested unit for $249.95. Part No. TA 1000C

T.V. INTERFACE



● Converts video to AM modulated RF, Channels 2 or 3. So powerful almost no tuning is required. On board regulated power supply makes this extremely stable. Rated very highly in Doctor Dobbs' Journal. Recommended by Apple ● Power required is 12 volts AC C.T., or +5 volts DC ● Board only $7.60 part No. 107, with parts $13.50 Part No. 107A

TAPE INTERFACE



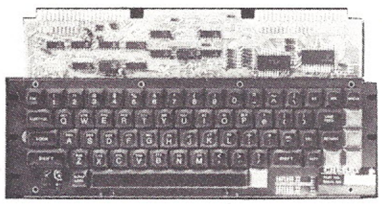
● Converts a low cost tape recorder to a digital recorder ● Works up to 1200 baud ● Digital in and out are TTL-serial ● Output of board connects to mic. in of recorder ● Earphone of recorder connects to input on board ● No coils ● Requires +5 volts, low power drain ● Board only $7.60 Part No. 111, with parts $29.95 Part No. 111A

ASCII KEYBOARD

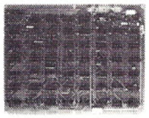
53 Keys popular ASR-33 format • Rugged G-10 P. C. Board • Tri-mode MOS encoding • Two-Key Rollover • MOS/DTL/TTL Compatible • Upper Case lockout • Data and Strobe inversion option • Three User Definable Keys • Low contact bounce • Selectable Parity • Custom Keycaps • George Risk Model 753. Requires +5, -12 volts. $59.95 Kit.

ASCII KEYBOARD

TTL & DTL compatible • Full 67 key array • Full 128 character ASCII output • Positive logic with outputs resting low • Data Strobe • Five user-definable spare keys • Standard 22 pin dual card edge connector • Requires +5VDC, 325 mA. Assembled & Tested. Cherry Pro Part No. P70-05AB. $119.95.

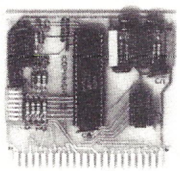


T.V. TYPEWRITER



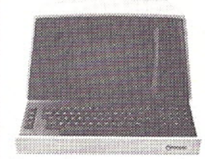
● Stand alone TVT ● 32 char/line, 16 lines, modifications for 64 char/line included ● Parallel ASCII (TTL) input ● Video output ● 1K on board memory ● Output for computer controlled curser ● Auto scroll ● Non-destructive curser ● Curser inputs: up, down, left, right, home, EOL, EOS ● Scroll up, down ● Requires +5 volts at 1.5 amps, and -12 volts at 30 mA ● All 7400, TTL chips ● Char. gen. 2513 ● Upper case only ● Board only $39.00 Part No. 106, with parts $145.00 Part No. 106A

UART & BAUD RATE GENERATOR



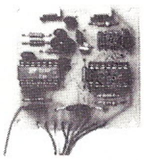
● Converts serial to parallel and parallel to serial ● Low cost on board baud rate generator ● Baud rates: 110, 150, 300, 600, 1200, and 2400 ● Low power drain +5 volts and -12 volts required ● TTL compatible ● All characters contain a start bit, 5 to 8 data bits, 1 or 2 stop bits, and either odd or even parity. ● All connections go to a 44 pin gold plated edge connector ● Board only $12.00 Part No. 101, with parts $35.00 Part No. 101A, 44 pin edge connector $4.00 Part No. 44P

SOROC IQ 120



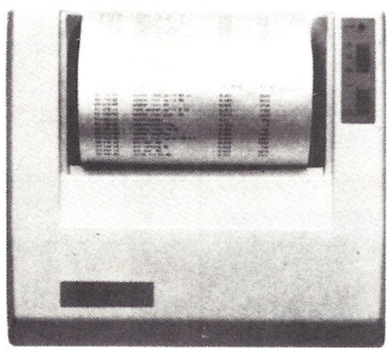
Upper/lower case display • Numeric keypad & cursor keys • Protected fields, ½ intensity display • RS 232 interface & aux. port. IQ120—$799.95 • IQ140 Detachable keyboard—$1199.95

MODEM



● Type 103 ● Full or half duplex ● Works up to 300 baud ● Originate or Answer ● Serial TTL input and output ● connect 8 Ω speaker and crystal mic. directly to board ● Requires +5 volts ● Board only $7.60 Part No. 109, with parts $29.95 Part No. 109A.

COMPRINT PRINTER



Printing Characteristics: 225 characters/second (170 lines/minute) throughput ● 9 horizontal x 12 vertical matrix ● 96 ASCII character set with upper and true lower case ● 80 characters/line ● 5.8 lines/inch
Buffer Memory: standard 256 bytes; ● optional; 2,048 bytes (buffer memory option designated as Model 912-2K), add $149.95.
Paper Requirements: electrosensitive type (aluminum coated) ● 8-1/2 inch width ● 3.7 inch max. (300 ft.) roll diameter.
Model 912-S Interfacing: serial interface RS232 and 20 mA current loop ● BAUD rates 110, 150, 300, 600, 1200, 2400 and 4800 are strap selectable.
Model 912-P Interfacing: parallel interface, IEEE-488 and 8 bit parallel (strobe/ acknowledge). Model 912-S, Part No. CPIA, 32118, $579.95. Model 912-P, Part No. CPIA, 32117, $559.95.

RS-32/TTL INTERFACE

● Converts TTL to RS-232, and converts RS-232 to TTL ● Two separate circuits ● Requires -12 and +12 volts ● All connections go to a 10 pin edge connector, kit $9.95 Part No. 232A 10 Pin edge connector $3.00 part No. 10P.

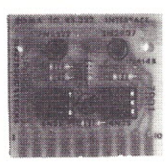
COMPUCOLOR II

With reg. keyboard
MOD3 8K $1595.95
MOD4 16K $1695.95
MOD 5 32K $1995.95
Now includes $250 more, worth of software and accessories with 101 key option add $134.95 with 117 key option add $179.95

44 BUS MOTHER BOARD

Has provisions for ten 44 pin (.156) connectors, spaced 3/4 of an inch apart. Pin 20 is connected to X, and 22 is connected to Z for power and ground All the other pins are connected in parallel. This board also has provisions for bypass capacitors. Board cost $15.00 Part No. 102. Connectors $3.00 each Part No. 44WP.

RS-232/20mA INTERFACE



This board has two passive, opto-isolated circuits. One converts RS-232 to 20mA, the other converts 20mA to RS-232. All connections go to a 10 pin edge connector. Requires +12 and -12 volts. Board only $9.95, part no. 7901, with parts $14.95 Part No. 7901A.

DC POWER SUPPLY

● Board supplies a regulated +5 volts at 3 amps., +12, -12, and -5 volts at 1 amp. ● Power required is 8 volts AC at 3 amps., and 24 volts AC C.T. at 1.5 amps. ● Board only $12.50 Part No. 6085, with parts excluding transformers $42.50 Part No. 6085A

To Order: Mention part no. description, and price. In USA shipping paid by us for orders accompanied by check or money order. We accept C.O.D. orders in the U. S. only, or a VISA or Master Charge no., expiration date, signature, phone no., shipping charges will be added. CA residents add 6.5% for tax. Outside USA add 10% for air mail postage and handling. Payment must be in U. S. dollars. Dealer inquiries invited. 24 hour order line (408) 448-0800 ✓47

Send for FREE Catalog . . . a big self-addressed envelope with 41¢ postage gets it fastest!

ELECTRONIC SYSTEMS Dept. KB P. O. Box 21638, San Jose, CA USA 95151

apple II Or APPLE II PLUS

16K $975.95, Extra 16K E.S. RAM installed $74.95, extra 32K E.S. RAM installed $148.95.

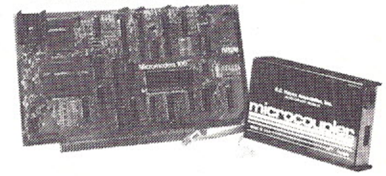
PET COMPUTER

With 16K & monitor - $ 795. Dual Disk Drive - $10 95

PARALLEL TRIAC OUTPUT BOARD FOR APPLE II

This board has 8 triacs capable of switching 110 volt 6 amp loads (660 watts per channel) or a total of 5280 watts. Board only $15.00 Part No. 210, with parts $119.95 Part No. 210A.

D.C. HAYES MICROMODEM

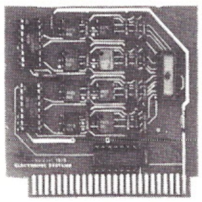


Fully S-100 bus compatible including 16-bit machines and 4 MHz processors. • Two software selectable Baud rates—300 Baud and a jumper selectable speed from 45 to 300 Baud. (110 standard). Supports originate and answer modes. • Direct-connect Microcoupler. This FCC-registered device provides direct access into your local telephone system, with none of the losses or distortions associated with acoustic couplers and without a telephone company supplied data access arrangement. • Auto-Answer/Auto-Call. The MICROMODEM 100 can automatically answer the phone and receive input; it can also dial a number automatically. • Automatic Reset and Disconnect. • Software compatible with the D.C. Hayes Associates 80-103A Data Communications Adapter. Micromodem-DCHA32625—$379.95

APPLE II PROTOTYPING CARD

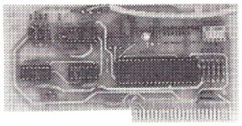
APPLE II HOBBY/ PROTOTYPING CARD Part No. 7907 $14.95

OPTO-ISOLATED PARALLEL INPUT BOARD FOR APPLE II



There are 8 inputs that can be driven from TTL logic or any 5 volt source. The circuit board can be plugged into any of the 8 sockets of your Apple II. It has a 16 pin socket for standard dip ribbon cable connection. Board only $15.00. Part No. 120, with parts $69.95. Part No. 120A.

APPLE II SERIAL I/O INTERFACE

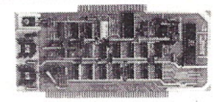


Baud rate is continuously adjustable from 0 to 30,000 • Plugs into any peripheral connector • Low current drain. RS-232 input and output • On board switch selectable 5 to 8 data bits, 1 or 2 stop bits, and parity or no parity either odd or even • Jumper selectable address • SOFTWARE • Input and Output routine from monitor or BASIC to teletype or other serial printer • Program for using an Apple II for a video or an intelligent terminal. Also can output in correspondence code to interface with some selectrics. • Also watches DTR • Board only $15.00 Part No. 2, with parts $42.00 Part No. 2A, assembled $62.00 Part No. 2C

REAL TIME 100,000 DAY CLOCK

MT. HARDWARE Double the utility of your S-100 bus computer with a real-time clock that keeps time in 100μS increments for over 273 years. Program events for the entire period with real time interrupts...without derailing the system. Maintain a log of computer usage, time and date transaction printouts, call up lists...virtually any activity where time is a factor. On-board battery backup. MHPX004—$249.95

TIDMA

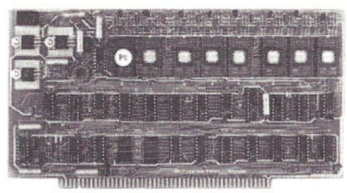


Tape Interface Direct Memory Access • Record and play programs without bootstrap loader (no prom) has FSK encoder/decoder for direct connections to low cost recorder at 1200 baud rate, and direct connections for inputs and outputs to a digital recorder at any baud rate • S-100 bus compatible • Board only $35.00 Part No. 112, with parts $110.00 Part No. 112A.

SYSTEM MONITOR

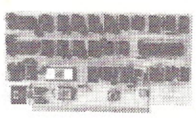
8080, 8085, or Z-80 System monitor for use with the TIDMA board. There is no need for the front panel. Complete with documentation $12.95.

8K EPROM PICEON



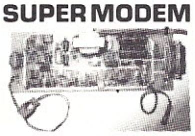
• Programs 2708's address relocation of each 4K of memory to any 4K boundary • Power on jump and reset jump option for "turnkey" systems and computers without a front panel • Program saver software in 1 2708 EPROM $25. Bare board $35 including custom coil, board with parts but no EPROMS $139, with 4 EPROMS $179, with 8 EPROMS $219.

VIDEO TERMINAL



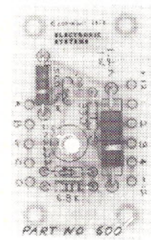
16 lines, 64 columns • Upper and lower case • 5x7 dot matrix • Serial RS-232 in and out with TTL parallel keyboard input • On board baud rate generator 75, 110, 150, 300, 600, & 1200 jumper selectable • Memory 1024 characters (7-21L02) • Video processor chip SFF96364 by Neculonic • Control characters (CR, LF, →, ←, ↑, ↓, non destructive cursor, CS, home, CL • White characters on black background or vice-versa • With the addition of a keyboard, video monitor or TV set with TV interface (part no. 107A) and power supply this is a complete stand alone terminal • also S-100 compatible • requires +16, & −16 VDC at 100mA, and 8VDC at 1A. Part No. 1000A $199.95 kit.

SUPER MODEM



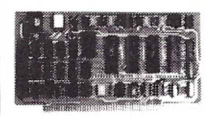
Orignate, RS-232 and 20 mA compatable, Full duplex, and half duplex, direct connect or acoustic coupled, on board power supply, carrier detect light, DB25 plug, 300 BAUD, Type 103 compatable frequencies, Bare board Part No. 2000, $19,95, Kit Part No. 2000A, $99.95.

RS-232/TTY INTERFACE



This board has two active circuits, one converts RS-232 to 20 mA, the other converts 20 mA to RS-232. Requires +12 and −12 volts. $9.95 Part No. 600A Kit.

SERIAL I/O



Four Serial I/O RS-232 ports. S-100 Bus, Software or jumper selectable baud rate (110, 300, 600, 1200, 2400, 4800, 9600, 19.2K), on board Xtal baud rate generator, Addressing, switch selectable, Parity or no parity (odd or even) switch selectable, 1 or 2 stop bits, 5 to 8 bits/character. Board only $29.95, Part No. 7908. With parts (kit) $199.95, Part No. 7908A.

WAMECO PRODUCTS

With ELECTRONIC SYSTEMS parts

FDC-1 FLOPPY CONTROLLER BOARD will drive shugart, pertek, remex 5" & 8" drives up to 8 drives, on board PROM with power boot up, will operate with CPM (not included). PCBD $42.95

FPB-1 Front Panel. (Finally) IMSAI size hex displays. Byte or instruction single step. PCBD $42.95

MEM-1A 8Kx8 fully buffered, S-100, uses 2102 type RAMS. PCBD .... $24.95, $168 Kit

QMB-12 MOTHER BOARD, 13 slot, terminated, S-100 board only .... $34.95 $89.95 Kit

CPU-1 8080A Processor board S-100 with 8 level vector interrupt PCBD .. $25.95 $89.95 Kit

RTC-1 Realtime clock board. Two independent interrupts. Software programmable. PCBD .... $25.95, $60.95 Kit

EPM-1 1702A 4K EPROM card PCBD .... $25.95 $49.95 with parts less EPROMS

EPM-2 2708/2716 16K/32K EPROM card PCBD .... $24.95 $49.95 with parts less EPROMS

QMB-9 MOTHER BOARD. Short Version of QMB-12. 9 Slots PCBD .... $30.95 $67.95 Kit

MEM-2 16Kx8 Fully Buffered 2114 Board PCBD .... $25.95, $269.95 Kit

16K EPROM

Uses 2708 EPROMS, memory speed selection provided, addressable anywhere in 65K of memory, can be shadowed in 4K increments. Board only $24.95 part no. 7902, with parts less EPROMs $49.95 part no. 7902A.

S-100 BUS ACTIVE TERMINATOR

Board only $14.95 Part No. 900, with parts $24.95 Part No. 900A

To Order: Mention part no. description, and price. In USA shipping paid by us for orders accompanied by check or money order. We accept C.O.D. orders in the U. S. only, or a VISA or Master Charge no., expiration date, signature, phone no., shipping charges will be added. CA residents add 6.5% for tax. Outside USA add 10% for air mail postage and handling. Payment must be in U. S. dollars. Dealer inquiries invited. 24 hour order line (408) 448-0800

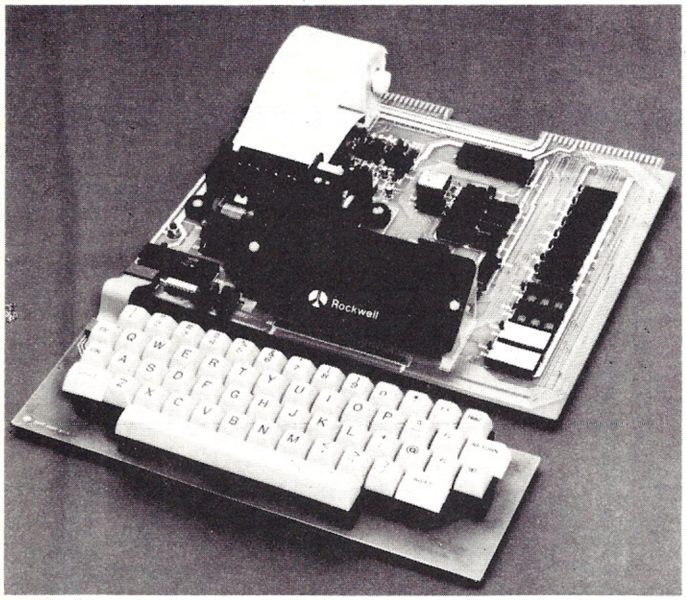
Send for FREE Catalog . . . a big self-addressed envelope with 41¢ postage gets it fastest!

ELECTRONIC SYSTEMS Dept. KB P. O. Box 21638, San Jose, CA USA 95151

✓47


MICROBYTE SYSTEM — 100
MICROBYTE
$499.95
System 100 includes:
• Microbyte Enclosure
• Microbyte 20 slot mother board with active termination
• IEEE S-100 compatible
• 25 amp heavy duty power supply
• Front console voltage monitors
• Custom air filter
• Corcom AC line filter
• 115/230v 50-60 Hz operation
• Hi CFM cooling fan
LP-2 Economy Logic Probe
For many applications, the LP-2 offers basic performance at a substantially lower price. Similar to the LP-1, but for slower circuits and without the Memory feature. Detects pulse events to 300 nsec and 1.5 MHz. High 300,000 Ohm input impedance guards against circuit loading.
$25.95
LP-3 High Speed Memory Probe
Our highest performance logic probe, the LP-3 detects pulses as narrow as 5 nsec (typical, tested at 10 nsec), pulse repetition rates to over 60 MHz. This is unquestionably the probe to use with today's higher speed logic systems. High input impedance. Other features similar to the LP-1. Includes LDA-5 ground wire (p 15).
$65.95
MSM5832 MICROPROCESSOR REAL-TIME CLOCK/CALENDAR
GENERAL DESCRIPTION
The MSM5832 is a monolithic, metal-gate CMOS integrated circuit that functions as a real time clock/calendar for use in bus-oriented microprocessor applications. The on-chip 32,768 Hz crystal controlled oscillator time base is counted down to provide addressable 4-bit I/O data of SECONDS, MINUTES, HOURS, DAY-OF-WEEK, DATE, MONTH, and YEAR. Data access is controlled by 4-bit address, chip select, read, write and hold inputs. Other functions include 12H/24H format selection, leap year identification and manual ± 30 second correction.
The MSM5832 normally operates from a 5 volt ± 5% supply. Battery back-up operation down to 2.2 volts allows continuation of time keeping when main power is off. One test input facilitates rapid testing of the time keeping operations. The MSM5832 is offered in an 18-lead dual-in-line plastic (RS suffix) package.
$745
FEATURES
• Microprocessor bus-oriented
• TIME MONTH DATE YEAR DAY OF WEEK
23 59 59 12 - 31 - 99 - 7
• 4-BIT DATA BUS
• 4-BIT ADDRESS
• Read, Write, Hold, Chip select inputs
• Interrupt signal outputs—1024, 1, 1/60, 1/3600 Hz
• 32.768 KHz crystal controlled operation
• Leap year register bit
• 12 or 24 hour format
• ± 30 second error correction
• Single 5 volt power supply
• Back-up battery operation to Vcc 2.2 V
• Low Power Dissipation
90 μw Max. at Vcc 3 V
2.5 mw Max. at Vcc 5 V
• High Density 300 mil 18-Pin Package
APPLE EXPANSION KIT
16K
Memory add-on with instructions
$54.95
Model 7811A for your APPLE
Arithmetic Processor
$369.00
APPLE GAME PORT REMOTE PLUG-IN EXPANDER with SELECTOR!
Allows continuous connection of any three of the Apple game port options.
$39.95
80-COLUMN IMPACT PRINTER
LOWEST IN PRICE—HIGHEST IN PERFORMANCE
*Compact Size
*80 Lines per Minute
*115/230VAC, 50 or 60 Hz.
*72, 80, 96, 120 or 132 Characters/Line
*Self-test Switch
*Auxiliary User Defined Character Set
*RS-232, 20ma, IEEE 488 and Centronics I/O
*16 Baud Rates to 19,200
$599.00
base2, inc.
EPROM ERASER BY SPECTOLINE
UP TO 6 EPROMS IN 19 MINUTES
6000 HOUR BULB LIFE
$49.95
MICROBYTE 19 SLOT BACKPLANE
• S-100 compatable
• IEEE standard
• Connectors have bifurcated, gold plated contacts
• Ground plane shielding for each signal
• Diode termination for ringing suppression
• Fully assembled and factory tested
$179.95
Model 2520A
Extender/Terminator
ACTIVE AND/OR DYNAMIC OPERATION
POWER LINES ALL FUSED
CCS $37.00
SOROC
IQ120
$749.00
DUST COVERS
Apple Disk 3.75
Apple Keyboard 8.75
TRS-80 Keyboard 7.75
TRS-80 Recorder 3.75
TRS-80 Single Disk 3.75
TRS-80 Double Disk 6.75
TRS-80 Model Keyboard 7.75
Pet Disk 7.75
Pet Terminal/Comp 15.75
Northstar Computer 11.75
Soroc Terminal 10.75
CSC
HI
PULSE
LO
LOGIC PROBE
$17.95
LOGIC PROBE KIT
S-100 16K ADD-ON BARE BOARD
WITH DOCUMENTATION AND DETAILED INSTRUCTION BOOK.
$28.95
CIRUIT SPECIALS
2708 1kx8 EPRM (450ns) 7.55
2716 1kx16 EPRM (450ns) 24.95
8038 VCO Waveform Gen. 2.55
4116 16kx1 Dyn RAM (300ns) 6.00
4116 16kx1 DYN RAM (200ns) 7.00
2114L 1kx4 Static RAM (450ns) 4.75
2114L 1kx4 Static RAM (250ns) 5.50
8212 I/O Port 3.25
8216 Bus Driver 1.99
74LS240 2.25
74LS241 2.25
74LS243 2.25
74LS244 2.25
74LS373 2.55
74LS374 2.55
74S138 3.95
APPLE II Computer with full 48K of memory!
$1099.00
Plexiglass cover as shown
$24.95
video 100
$139.00
Leedex Corp.
12" BLACK & WHITE LOW COST VIDEO MONITOR
IQ140-$1199.00
16K STATIC RAM
S-100 MEMORY ADD-ON
** FULLY STATIC OPERATION
** USES 2114 TYPE STATIC RAMSMS
** +8 VDC INPUT AT LESS THAN 2 AMPS
** BANK SELECT AVAILABLE BY BANK PORT AND BANK BYTE
** PHANTOM LINE CAPABILITY
** ADDRESSABLE IN 4K BLOCKS IN 4K INCREMENTS
** LED INDICATORS FOR BOARD/BANK
** SOLDER MASK ON BOTH SIDES OF BOARD
ASSEMBLED & TESTED
$269.00
California Computer Systems
KIT
TRS 80
16 K Memory Add-On
$64.95
with jumpers and instructions
CONCORD
COMPUTER COMPONENTS
297
1971 SOUTH STATE COLLEGE • ANAHEIM, CA. 92806
VISA-MASTERCHARGE
CHECK OR M.O.
NO C.O.D.
(714) 937-0637
We stock and sell over 12,000 types of semi-conductors
MINIMUM ORDER-$10.00
ADD $1.50 FOR FRT.
CAL. RES. ADD 6%

AIM 65 BY ROCKWELL INTERNATIONAL



AIM 65 is fully assembled, tested and warranted. With the addition of a low cost, readily available power supply, it's ready to start working for you.

AIM 65 features on-board thermal printer and alphanumeric display, and a terminal-style keyboard. It has an addressing capability up to 65K bytes, and comes with a user-dedicated 1K or 4K RAM. Two installed 4K ROMS hold a powerful Advanced Interface Monitor program, and three spare sockets are included to expand on-board ROM or PROM up to 20K bytes.

An Application Connector provides for attaching a TTY and one or two audio cassette recorders, and gives external access to the user-dedicated general purpose I/O lines.

Also included as standard are a comprehensive AIM 65 User's Manual, a handy pocket reference card, an R6500 Hardware Manual, an R6500 Programming Manual and an AIM 65 schematic.

AIM 65 is packaged on two compact modules. The circuit module is 12 inches wide and 10 inches long, the keyboard module is 12 inches wide and 4 inches long. They are connected by a detachable cable.

THERMAL PRINTER
Most desired feature on low-cost microcomputer systems . . .
- Wide 20-column printout
- Versatile 5 x 7 dot matrix format
- Complete 64-character ASCII alphanumeric format
- Fast 120 lines per minute
- Quite thermal operation
- Proven reliability

FULL-SIZE ALPHANUMERIC KEYBOARD
Provides compatibility with system terminals . . .
- Standard 54 key, terminal-style layout
- 26 alphabetic characters
- 10 numeric characters
- 22 special characters
- 9 control functions
- 3 user-defined functions

TRUE ALPHANUMERIC DISPLAY
Provides legible and lengthy display . . .
- 20 characters wide
- 16-segment characters
- High contrast monolithic characters
- Complete 64-character ASCII alphanumeric format

PROVEN R6500 MICROCOMPUTER SYSTEM DEVICES
Reliable, high performance NMOS technology . . .
- R6502 Central Processing Unit (CPU), operating at 1 MHz. Has 65K address capability, 13 addressing modes and true index capability. Simple but powerful 56 instructions.
- Read/Write Memory, using R2114 Static RAM devices. Available in 1K byte and 4K byte versions.
- 8K Monitor Program Memory, using R2332 Static ROM devices. Has sockets to accept additional 2332 ROM or 2532 PROM devices, to expand on-board Program memory up to 20K bytes.
- R6532 RAM-Input/Output-Timer (RIOT) combination device. Multipurpose circuit for AIM 65 Monitor functions.
- Two R6522 Versatile Interface Adapter (VIA) devices, which support AIM 65 and user functions. Each VIA has two parallel and one serial 8-bit, bidirectional I/O ports, two 2-bit peripheral handshake control lines and two fully-programmable 16-bit interval timer/event counters.

BUILT-IN EXPANSION CAPABILITY
- 44-Pin Application Connector for peripheral add-ons
- 44-Pin Expansion Connector has full system bus
- Both connectors are KIM-1 compatible

TTY AND AUDIO CASSETTE INTERFACES
Standard interface to low-cost peripherals . . .
- 20 ma. current loop TTY interface
- Interface for two audio cassette recorders
- Two audio cassette formats: ASCII KIM-1 compatible and binary, blocked file assembler compatible

ROM RESIDENT ADVANCED INTERACTIVE MONITOR
Advanced features found only on larger systems . . .
- Monitor-generated prompts
- Single keystroke commands
- Address independent data entry
- Debug aids
- Error messages
- Option and user interface linkage

ADVANCED INTERACTIVE MONITOR COMMANDS
- Major Function Entry
- Instruction Entry and Disassembly
- Display/Alter Registers and Memory
- Manipulate Breakpoints
- Control Instruction/Trace
- Control Peripheral Devices
- Call User-Defined Functions
- Comprehensive Text Editor

LOW COST PLUG-IN ROM OPTIONS
- 4K Assembler—symbolic, two-pass , **A65-010 $79.00**
- 8K BASIC Interpreter **A65-020 $99.00**

POWER SUPPLY SPECIFICATIONS
- +5 VDC ± 5% regulated @ 2.0 amps (max)
- +24 VDC ± 15% unregulated @ 2.5 amps (peak) 0.5 amps average

PRICE: $389.00 (1K RAM) $439.00 (4K RAM)

Plus $4.00 UPS (shipped in U.S. must give ***street*** address), $10 parcel post to APO's, FPO's, Alaska, Hawaii, Canada, $25 air mail to all other countries
AIM 65 USER MANUAL $5.00 plus $1.50 shipping & handling.
We manufacture a complete line of high quality expansion boards. Use reader service card to be added to our mailing list, or U.S. residents send $1.00 (International send $3.00 U.S.) for airmail delivery of our complete catalog.

RNB ENTERPRISES INCORPORATED ✓52

2967 W. Fairmount Avenue
Phoenix AZ. 85017
(602)265-7564

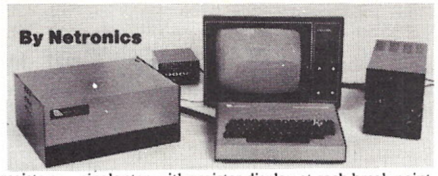
master charge VISA

✓ Reader Service index—page 241

Start Computing For Just $129.95 With An 8085-Based Professional Computer Kit—

Explorer/85

100% compatible with all 8080A and 8085 software & development tools!



No matter what your future computing plans may be, Level "A"—at $129.95—is your starting point.

Starting at just $129.95 for a Level "A" operating system, you can now build the exact computer you want. Explorer/85 can be your beginner's system, OEM controller, or IBM-formatted 8" disk small business system... yet you're never forced to spend a penny for a component or feature you don't want and you can expand in small, affordable steps!

Now, for just $129.95, you can own the first level of a fully expandable computer with professional capabilities—a computer which features the advanced Intel 8085 cpu, *thereby giving you immediate access to all software and development tools that exist for both the 8085 and its 8080A predecessor (they are 100% software compatible)*—a computer which features onboard S-100 bus expansion—plus instant conversion to mass storage disk memory with either 5-1/4" diskettes or standard IBM-formatted 8" disks.

For just $129.95 (plus the cost of a power supply, keyboard/terminal and RF modulator, if you don't have them already), Explorer/85 lets you begin *computing* on a significant level... *applying* the principles discussed in leading computer magazines... *developing* "state of the art" computer solutions for both the industrial and leisure environment.

Level "A" Specifications

Explorer/85's Level "A" system features the advanced Intel 8085 cpu, an 8355 ROM with 2k deluxe monitor/operating system, and an 8155 ROM-I/O—all on a single motherboard with room for RAM/ROM/PROM/EPROM and S-100 expansion, *plus* generous prototyping space.

(Level "A" makes a perfect OEM controller for industrial applications and is available in a special Hex Version which can be programmed using the Netronics Hex Keypad/Display.)

Level "A" at $129.95 is a complete operating system, perfect for beginners, hobbiests, or industrial controller use.

PC Board: glass epoxy, plated through holes with solder mask • **I/O:** provisions for 25-pin (DB25) connector for terminal serial I/O, which can also support a paper tape reader ...provision for 24-pin DIP socket for hex keyboard/display... cassette tape recorder input... cassette tape recorder output... cassette tape control output... speaker output... LED output indicator on SOD (serial output) line... printer interface (less drivers)... total of four 8-bit plus one 6-bit I/O ports • **Crystal Frequency:** 6.144 MHz • **Control Switches:** reset and user (RST 7.5) interrupt... additional provisions for RST 5.5, 6.5 and TRAP interrupts onboard • **Counter/Timer:** programmable, 14-bit binary • **System RAM:** 256 bytes located at F800, ideal for smaller systems and for use as an isolated stack area in expanded systems... RAM expandable to 64k via S-100 bus or 4K on motherboard.

System Monitor (Terminal Version): 2k bytes of deluxe system monitor ROM located at F000 leaving 0000 free for user RAM/ROM. Features include tape load with labeling ... tape dump with labeling... examine/change contents of memory ... insert data... warm start... examine and change all registers... single step with register display at each break point, a debugging/training feature... go to execution address... move blocks of memory from one location to another... fill blocks of memory with a constant... display blocks of memory ... automatic baud rate selection... variable display line length control (1-255 characters/line)... channelized I/O monitor routine with 8-bit parallel output for high speed printer... serial console in and console out channel so that monitor can communicate with I/O ports.

System Monitor (Hex Version): Tape load with labeling... tape dump with labeling... examine/change contents of memory... insert data... warm start... examine and change all registers... single step with register display at each break point ...go to execution address. Level "A" in the *Hex Version* makes a perfect controller for industrial applications and can be programmed using the Netronics Hex Keypad/Display.

Hex Keypad/Display.

Hex Keypad/Display Specifications

Calculator type keypad with 24 system defined and 16 user defined keys. 6 digit calculator type display which displays full address plus data as well as register and status information.

Level "B" Specifications

Level "B" provides the S-100 signals plus buffers/drivers to support up to six S-100 bus boards and includes: address decoding for onboard 4k RAM expansion select-able in 4k blocks... address decoding for onboard 8k EPROM expansion selectable in 8k blocks... address and data bus drivers for onboard expansion... wait state generator (jumper selectable), to allow the use of slower memories... two separate 5 volt regulators.

Explorer/85 with Level "C" card cage.

Level "C" Specifications

Level "C" expands Explorer's motherboard with a card cage, allowing you to plug up to six S-100 cards directly into the motherboard. Both cage and cards are neatly contained inside Explorer's deluxe steel cabinet. Level "C" includes a sheet metal superstructure, a 5-card gold plated S-100 extension PC board which plugs into the motherboard. Just add required number of S-100 connectors

Level "D" Specifications

Level "D" provides 4k or RAM, power supply regulation, filtering decoupling components and sockets to expand your Explorer/85 memory to 4k (plus the original 256 bytes located in the 8155A). The static RAM can be located anywhere from 0000 to EFFF in 4k blocks.

Level "E" Specifications

Level "E" adds sockets for 8k of EPROM to use the popular Intel 2716 or the TI 2516. It includes all sockets, power supply regulator, heat sink, filtering and decoupling components. Sockets may also be used for soon to be available RAM IC's (allowing for up to 12k of onboard RAM).

Order A Coordinated Explorer/85 Applications Pak!

Experimenter's Pak (SAVE $12.50)—Buy Level "A" and Hex Keypad/Display for **$199.90** and get FREE Intel 8085 user's manual plus FREE postage & handling!

Student Pak (SAVE $24.45)—Buy Level "A," ASCII Keyboard/Computer Terminal, and Power Supply for **$319.85** and get FREE RF Modulator plus FREE Intel 8085 user's manual plus FREE postage & handling!

Engineering Pak (SAVE $41.00)—Buy Levels "A," "B," "C," "D," and "E" with Power Supply, ASCII Keyboard/Computer Terminal, and six S-100 Bus Connectors for **$514.75** and get 10 FREE computer grade cassette tapes plus FREE 8085 user's manual plus FREE postage & handling!

Business Pak (SAVE $89.95)—Buy Explorer/85 Levels "A," "B," and "C" (with cabinet), Power Supply, ASCII Keyboard/Computer Terminal (with cabinet), 16k RAM, 12" Video Monitor, North Star 5-1/4" Disk Drive (includes North Star BASIC) with power supply and cabinet, all for just **$1599.40** and get 10 FREE 5-1/4" minidiskettes ($49.95 value) plus FREE 8085 user's manual plus FREE postage & handling!

Continental U.S.A. Credit Card Buyers Outside Connecticut

CALL TOLL FREE 800-243-7428

To Order From Connecticut Or For Technical Assistance, Etc. **Call (203) 354-9375**

Netronics R&D Ltd.
333 Litchfield Road, New Milford, CT 06776

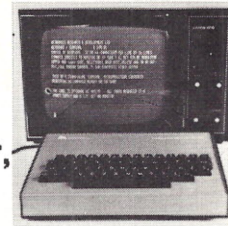
Please send the items checked below—

- ☐ **Explorer/85 Level "A" Kit** (ASCII Version), **$129.95** plus $3 p&h.
- ☐ **Explorer/85 Level "A" Kit** (Hex Version), **$129.95** plus $3 p&h.
- ☐ **8k Microsoft BASIC** on cassette tape, **$64.95** postpaid.
- ☐ **8k Microsoft BASIC in ROM Kit** (requires Levels "B," "D," and "E"), **$99.95** plus $2 p&h.
- ☐ **Level "B" (S-100) Kit, $49.95** plus $2 p&h.
- ☐ **Level "C" (S-100 6-card expander) Kit, $39.95** plus $2 p&h.
- ☐ **Level "D" (4k RAM) Kit, $69.95** plus $2 p&h.
- ☐ **Level "E" (EPROM/ROM) Kit, $5.95** plus 50¢ p&h.
- ☐ **Deluxe Steel Cabinet** for Explorer/85, **$49.95** plus $3 p&h.
- ☐ **ASCII Keyboard/Computer Terminal Kit** (features a full 128 character set, upper & lower case, full cursor control, 75 ohm video output convertible to baudot output, selectable baud rate, RS232-C or 20 ma. I/O, 32 or 64 character by 16 line formats, and can be used with either a CRT monitor or a TV set (if you have an RF modulator), **$149.95** plus $2.50 p&h.
- ☐ **Hex Keypad/Display Kit, $69.95** plus $2 p&h.
- ☐ **Deluxe Steel Cabinet** for ASCII Keyboard/Terminal, **$19.95** plus $2.50 p&h.
- ☐ **Power Supply Kit** (± 8V @ 5 amps) in deluxe steel cabinet, **$39.95** plus $2 p&h.
- ☐ **Gold Plated S-100 Bus Connectors, $4.85** each, postpaid.
- ☐ **RF Modulator Kit** (allows you to use your TV set as a monitor), **$8.95** postpaid.
- ☐ **16k RAM Kit** (S-100 Board expands to 64k), **$199.95** plus $2 p&h.
- ☐ **32k RAM Kit, $329.95** plus $2 p&h.
- ☐ **48K RAM Kit, $459.95** plus $2 p&h.
- ☐ **64k RAM Kit, $589.95** plus $2 p&h.
- ☐ **16k RAM Expansion Kit** (to expand any of the above up to 64k), **$139.95** plus $2 p&h each.
- ☐ **Intel 8085 cpu User's Manual, $7.50** postpaid.
- ☐ **Special Computer Grade Cassette Tapes, $1.90** each or **3 for $5,** postpaid.
- ☐ **12" Video Monitor** (10 MHz bandwidth), **$139.95** plus $5 p&h.
- ☐ **North Star Double Density Floppy Disk Kit** (One Drive) for Explorer/85 (includes 3 drive S-100 controller, DOS, and extended BASIC with personalized disk operating system—just plug it in and you're up and running!), **$699.95** plus $5 p&h.
- ☐ **Power Supply Kit** for North Star Disk Drive, **$39.95** plus $2 p&h.
- ☐ **Deluxe Case** for North Star Disk Drive, **$39.95** plus $2 p&h.
- ☐ **Experimenter's Pak** (see above), **$199.90** postpaid.
- ☐ **Student Pak** (see above), **$319.85** postpaid.
- ☐ **Engineering Pak** (see above), **$514.75** postpaid.
- ☐ **Business Pak** (see above), **$1599.40** postpaid.

Total Enclosed $ ______
(Conn. res. add sales tax) By—
☐ Personal Check ☐ M.O./Cashier's Check ☐ Visa ☐ Master Charge
(Bank # ____)

Acct. # ______
Signature ______ Exp. Date ____
Print Name ______
Address ______
City ______
State ______ Zip ______

☐ **Send Me Information**



By Netronics

ASCII/BAUDOT, STAND ALONE

Computer Terminal

COMPLETE FOR ONLY... $149.95

The Netronics ASCII/BAUDOT Computer Terminal Kit is a microprocessor-controlled, stand alone keyboard/terminal requring no computer memory or software. It allows the use of either a 64 or 32 character by 16 line professional display format with selectable baud rate, RS232-C or 20 ma. output, full cursor control and 75 ohm composite video output.

The keyboard follows the standard typewriter configuration and generates the entire 128 character ASCII upper/lower case set with 96 printable characters. Features include onboard regulators, selectable parity, shift lock key, alpha lock jumper, a drive capability of one TTY load, and the ability to mate directly with almost any computer, including the new Explorer/85 and ELF products by Netronics.

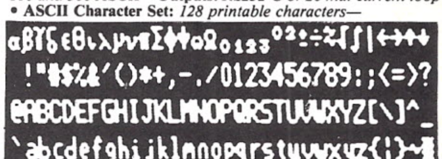
The Computer Terminal requires no I/O mapping and includes 1k of memory, character generator, 2 key rollover, processor controlled cursor control, parallel ASCII/BAUDOT to serial conversion and serial to video processing—fully crystal controlled for superb accuracy. PC boards are the highest quality glass epoxy for the ultimate in reliability and long life.

VIDEO DISPLAY SPECIFICATIONS

The heart of the Netronics Computer Terminal is the microprocessor-controlled Netronics Video Display Board (VID) which allows the terminal to utilize either a parallel ASCII or BAUDOT signal source. The VID converts the parallel data to serial data which is then formatted to either RS232-C or 20 ma. current loop output, which can be connected to the serial I/O on your computer or other interface, i.e., Modem.

When connected to a computer, the computer must echo the character received. This data is received by the VID which processes the information, converting to data to video suitable to be displayed on a TV set (using an RF modulator) or on a video monitor. The VID generates the cursor, horizontal and vertical sync pulses and performs the housekeeping relative to which character and where it is to be displayed on the screen.

Video Output: *1.5 P/P into 75 ohm (EIA RS-170)* • **Baud Rate:** *110 and 300 ASCII* • **Outputs:** *RS232-C or 20 ma. current loop* • **ASCII Character Set:** *128 printable characters—*



BAUDOT Character Set: *A B C D E F G H I J K L M N O P Q R S T U V W X Y Z - ? : * 3 $ # ( ) . , 9 0 1 4 ! 5 7 ; 2 / 6 8* • **Cursor Modes:** *Home, Backspace, Horizontal Tab, Line Feed, Vertical Tab, Carriage Return. Two special cursor sequences are provided for absolute and relative X-Y cursor addressing* • **Cursor Control:** *Erase, End of Line, Erase of Screen, Form Feed, Delete* • **Monitor Operation:** *50 or 60Hz (jumper selectable.*

Continental U.S.A. Credit Card Buyers Outside Connecticut

CALL TOLL FREE 800-243-7428

To Order From Connecticut Or For Technical Assistance, Etc. **Call (203) 354-9375**

Netronics R&D Ltd., Dept. K-7
333 Litchfield Road, New Milford, CT 06776

Please send the items checked below—

- ☐ **Netronics Stand Alone ASCII Keyboard/Computer Terminal Kit, $149.95** plus $3.00 postage & handling.
- ☐ **Deluxe Steel Cabinet** for Netronics Keyboard/Terminal In Blue/Black Finish, **$19.95** plus $2.50 postage and handling.
- ☐ **Video Display Board Kit** alone (less keyboard), **$89.95** plus $3 postage & handling.
- ☐ **12" Video Monitor** (10 MHz bandwidth) fully assembled and tested, **$139.95** plus $5 postage and handling.
- ☐ **RF Modulator Kit** (to use your TV set for a monitor), **$8.95** postpaid.
- ☐ **5 amp Power Supply Kit** In Deluxe Steel Cabinet (± 8VDC @ 5 amps, plus 6-8 VAC), **$39.95** plus $2 postage & handling.

Total Enclosed (Conn. res. add sales tax) $ ______
By—
☐ Personal Check ☐ Cashiers Check/Money Order
☐ **Visa** ☐ **Master Charge** (Bank # ____)

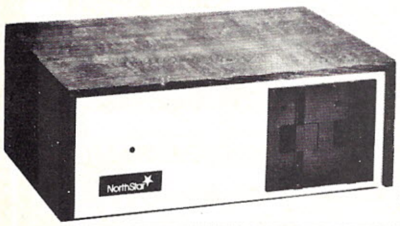
Acct. # ______
Signature ______ Exp. Date ______
Print Name ______
Address ______
City ______
State ______ Zip ______

☐ **Send Me More Information**

If North Star or Cromemco offer it . . .

WE HAVE IT!!

Immediate Delivery at Discount Prices



NORTH STAR
Horizon® 2
32K Double Density
Assembled and Tested
List $3095
ONLY $2619

North Star KIT products have been discontinued. MiniMicroMart HAS INVENTORY of most items!

KITS		ASSEMBLED	
HORIZON 1 16K, DD	**$1474**	**HORIZON 1**, DD	**$2279**
32K, DD, List $1999	**1684**	32K, QD, List $2995	**2539**
32K, QD, List $2199	**1869**	**HORIZON 2**, 32K, DD	**$2619**
HORIZON 2, 16K, DD	**$1824**	32K, QD, List $3595	**3049**
32K, DD, List $2399	**2034**	48K, DD, List $3590	**3039**
32K, QD, List $2779	**2359**	48K, QD, List $4090	**3469**
		64K, DD, List $3830	**3239**
		64K, QD, List $4330	**3669**

NORTH STAR APPLICATIONS SOFTWARE

(Exclusive for use with North Star Disk Systems — specify Double or Quad Density)

NORTHWORD, List $399	**$339**
MAILMANAGER, List $299	**249**
INFOMANAGER, List $499	**419**
GENERALLEDGER, List $999	**799**
ACCOUNTSRECEIVABLE, List $599	**499**
ACCOUNTSPAYABLE, List $599	**499**

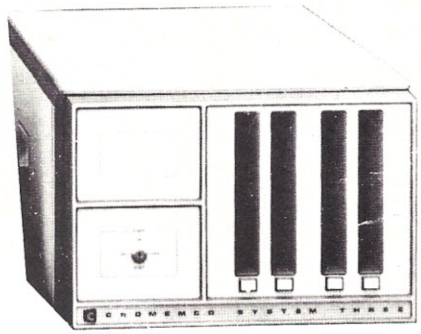
NORTH STAR HARD DISK HD-18

18 megabytes, plugs into parallel port of North Star Horizon. Utilizes tried-and-proven 14" Century Data Marksman. List $4999. **OUR PRICE $4199**

NORTH STAR MDS-A — Double (or Quad) Density Disk System, Kit, List $799 . **OUR PRICE $669**
Assembled and Tested, List $899 **SPECIAL $719**

NORTH STAR MEMORY BOARDS

16K Dynamic RAM (RAM-16-A/A), Assembled, List $499	**$420**
Kit, List $449	**SPECIAL $299**
32K (RAM-32/A), Assembled, List $739	**$620**
Kit, List $669	**ONLY $499**



CROMEMCO
System 3
with 64K of RAM
List $6990
OUR PRICE
$5890

CROMEMCO SYSTEM 2 — Now features dual-sided drives — double the capacity. Similar to System 3, except features dual, double-sided mini floppy disk drives. List $3990 .... **ONLY $3390**

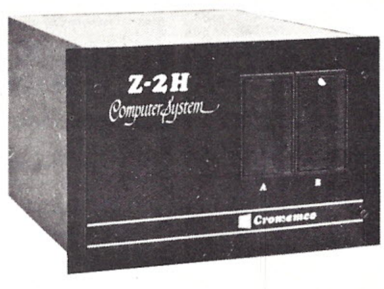
Z-2 COMPUTER SYSTEM (can be rack mounted), List $995 .... **$845**

SINGLE CARD COMPUTER — SCC-W
4 MHz. List $450 .... **$382**

***NEW* COLOR GRAPHICS INTERFACE — SDI**
List $595 .... ***OUR PRICE ONLY $505***

CROMEMCO HDD – 11/22-megabyte Hard Disk for use with existing systems. DMA controller. Transfer rate of 5.6 megabytes/second.
HDD-11, List $6995 ...... **OUR PRICE ONLY $5939**
HDD-22, List $11,995 .... **$10,189**

CROMEMCO Z-2H Full 11-megabyte Hard Disk system. Fast Z-80A 4 MHz processor, two floppy disk drives, 64K RAM memory, RS232 special interface, printer interface, and extensive software available. List $9995



OUR PRICE $8489

SHIPPING AND INSURANCE: Add $15 or Horizons, $2.50 for Boards and Software. Hard Disk Systems and Cromemco systems shipped freight collect. Advertised prices are for prepaid orders. Credit card and C.O.D. 2% higher. Deposit may be required on C.O.D. All prices subject to change and offers subject to withdrawal without notice.

— WRITE FOR FREE CATALOG —

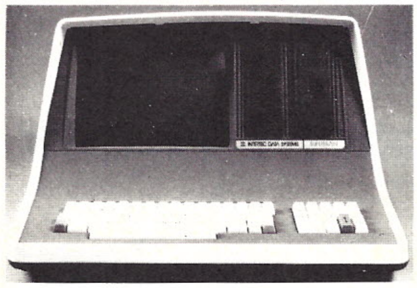
226 **MiniMicroMart, Inc.**

1618 James Street, Syracuse, NY 13203 (315) 422-4467 TWX 710-541-0431

master charge THE INTERBANK CARD
VISA

Try to beat our prices!

SUPERBRAIN® by Intertec



Self-contained computer with dual disks and two RS232C ports. Complete with CP/M® 2.2 and BASIC.

32K Double Density, List $2995 . **$2685**
64K Double Density, List $3345 .... **$2883**
64K MiniMicroMart-upgraded to Quad Density .... ***SPECIAL $3395***

VIDEO TERMINALS

INTERTUBE II, List $995 ...... **ONLY $ 799**
SOROC 120, List $995 ...... ***SPECIAL $ 729***
IQ140, List $1495 .... ***SPECIAL $1149***
PERKIN-ELMER 550, List $997 ...... **$ 799**
with anti-glare screen, $1027 .... **$ 829**
HAZELTINE 1410, List $900 .... **$ 749**
1420 .... **$ 849**
1500, List $1225 .... **$ 879**
1510, List $1395 .... **$1089**
1520, List $1650 .... **$1389**
ADDS R-20, List $995 .... **$ 945**
LEAR SIEGLER ADM3A, Assembled .. **$ 849**
TELEVIDEO 912 B or C .... **$ 789**
920 B or C .... **$ 849**

PRINTERS

ANADEX DP-8000 .... ***SPECIAL $ 749***
INTEGRAL DATA IP-125 ***NOW ONLY $ 689***
IP-125 w/1210 option, $838 ..... **NOW $ 724**
IP-225 w/1210 & 1250 op., List $988 ... **$ 834**
IP-225 w/tractor, 1210, 1250, 1221 (2K Buffer), 1241 (graphics) ... **NOW $ 899**
PAPER TIGER IDS-440, List $995 ..... **$ 895**
w/graphics op., incl. buffer, $1195 ... **$ 989**
NEC Spinwriters .... **Call for Price**
TELETYPE 43 KSR **$1087**
CENTRONICS
730-1 parallel interface .... ***NEW LOW $679***
737 parallel interface .. ***SUPER VALUE $849***
779 w/Tractor, List $1350 .... **$1049**
702 w/Tractor, VFU, List $2480 ...... **$1995**
703 w/Tractor, VFU, List $2975 ...... **$2395**
704 w/Tractor, VFU, List $2350 ...... **$1995**
T.I. 810 Basic, List $1895 .... **$1695**
810/serial & Centronics-style parallel interface, List $1940 .... **$1735**
810 w/full ASCII (U/LC), Vertical Forms Control, Compressed Print .. **$1895**
COMPRINT 912 w/parallel interface .. **$ 559**
912 w/serial interface, List $699 ...... **$ 589**
AXIOM IMP I .... **Call for Price**
MICROTEK, List $750 .... **$ 675**
OKIDATA Microline 80, List $949 .... **$ 649**
Tractor Feed Option .... **$ 109**

NORTH STAR HORIZON

HORIZON 1 KITS
16K, Double Density, List $1749 .... **$1474**
32K, Double Density, List $1999 ...... **$1684**
32K, Quad Density, List $2199 .... **$1869**

HORIZON 1 ASSEMBLED & TESTED
32K, Double Density, List $2695 .... **$2279**
32K, Quad Density, List $2995 .... **$2539**

HORIZON 2 KITS
16K, Double Density, List $2149 ...... **$1824**
32K, Double Density, List 2399 .... **$2034**
32K, Quad Density, List 2779 .... **$2359**

HORIZON 2 ASSEMBLED & TESTED
32K, Double Density, List $3095 ...... **$2619**
32K, Quad Density, List $3595 .... **$3049**
48K, Double Density, List $3590 ...... **$3039**
48K, Quad Density, List $4090 .... **$3469**
64K, Double Density, List $3830 ...... **$3239**
64K, Quad Density, List $4330 .... **$3669**

FLOPPY DISK SYSTEMS

NORTH STAR MDS-A
Assembled, List $899 ...... ***SPECIAL $ 719***
Kit Version, List $799 .... **$ 669**
MORROW THINKER TOYS® Discus 2D, List $1199 .... **OUR PRICE $1019***
Discus 2D, dual-drive, List $1994 ..... **$1694***
Discus 2+2, A&T, List $1549 .... **$1319***
Dual Discus 2+2, A&T, List $2748 ... **$2335***
*Now includes CP/M® 2.2
MICROMATION Megabox, DD w/ 8" drives, 1-megabyte, List $2295 ..... **$1949**
2-megabyte, List $3095 .... **$2629**
MICROPOLIS 1041 MacroFloppy®
w/enclosure (no P.S.), List $695 ...... **$ 625**
1042 MacroFloppy w/case & AC P.S. . **$ 709**
1053 Dual MetaFloppy® , List $1895 ... **$1695**

VIDEO BOARDS

I/O Mapped
SD COMPUTER VDB-8024, kit, List $370 **$299**
Assembled, List $470 .... **$ 399**
XITEX SCT-100K, Kit .... **ONLY $154.95**
SCT-100A Assembled **$174.95**
SSM VB2 I/O, Kit, List $169 .... **$ 144**
Assembled & Tested, List $234 ...... **$ 199**
Memory Mapped
SSM VB1B, 16x64, Kit, List $155 ..... **$ 132**
Assembled & Tested, List $210 ...... **$ 178**
SSM VB3, 80-Char., 4MHz, Kit, List $399 **$ 339**
4 MHz, A&T, List $464 .... **$ 394**
INTERSYSTEMS, 16x64, A&T, List $165 **$149**

ESCON CONVERSION FOR IBM SELECTRIC

Complete w/microprocessor controller and power supply. Factory built. User installs solenoid assembly or it can be done at Escon factory at nominal cost.

Parallel (TRS-80, Sorcerer, etc.), $575 **$514**
RS232 Standard Serial, List $599 ...... **534**
IEEE-488 (for PET), List $660 .... **584**
TRS-80 Cable **25**

CPU BOARDS
(assembled unless noted)

NORTH STAR Z80A (ZPB-A/A), $299 **$254**
CROMEMCO 4 MHz (ZPU-W), List $395 **$335**
4 MHz (SCC-W), List $450 .... **$382**
INTERSYSTEMS (formerly Ithaca Audio)
new Series II Z-80, 4 MHz, List $395 ... **$349**
SSM CB1 8080 A&T, List $219 .... **$186**
CB1A Kit, List $159 .... **$135**
CB2 Z-80, A&T, List $275 .... **$234**
CB2 Kit, List $210 .... **$179**
DELTA Z-80, with I/O **$289**
SD SBC-100, List $350 .... **$298**
SBC-100 Kit, List $295 .... **$250**
SBC-200, List $400 .... **$332**
SBC-200 Kit, List $320 .... **$272**

MEMORY BOARDS

IMAGINE!
32K SD ExpandoRAM Kit
ONLY $249
$159 without RAM chips

NORTH STAR 16K Dynamic RAM Board, A&T (RAM-16-A/A), List $499 .... **$420**
16K Kit Version, List $449 ... **SPECIAL $299**
32K A&T (RAM-32/A), List $739 ...... **$620**
32K Kit, List $669 .... **SPECIAL $499**
CROMEMCO (16KZ-W), List $595 ..... **$495**
64KZ-W, List $1795 .... **1485**
MEASUREMENT SYSTEMS & CONTROLS
(Guaranteed performance, incl. labor/parts 1 yr)
DM6400 64K Board w/all 64K, $795 ... **$659**
DM4800 with 48K, List $695 .... **$589**
DM3200 with 32K, List $595 .... **$509**
DMB6400 64K Board w/all 64K **$859**
DMB4800 with 48K **$789**
MORROW SuperRAM — all static, all A&T
16K, 4 MHz or 2 MHz, List $349 .... **$299**
32K, 4 MHz, List $699 .... **$629**
16K Memory Master, List $399 .... **$339**
24K Memory Master, List $549 .... **$465**
INTERSYSTEMS (formerly Ithaca Audio)
8K Static 2 MHz, A&T, List $165 ...... **$149**
8K Static 4 MHz, A&T, List $195 ...... **$176**
16K Static 2 MHz, A&T, List $475 ..... **$427**
16K Static 4 MHz, A&T, List $495 ..... **$445**
64K Dynamic, List $995 .... **$895**
CALIFORNIA COMPUTER
16K Static, A&T, List $349.95 .... **$259**

FLOPPY DISK CONTROLLER BOARDS

MORROW Disk Jockey 1, A&T ($213) . **$189**
Disk Jockey 2D, A&T, List $479 .... **$429**
SD Versafloppy 1, Kit, List $250 .... **$212**
Versafloppy II, DD Kit, List $350 ...... **$278**
Versafloppy II, DD, A&T, List $430 .... **$360**
DELTA double density A&T ($385) **$345**
CONDUCTOR, double density A&T ... **$269**
INTERSYSTEMS FDC-2, A&T, $495 ... **$439**
MICROMATION Doubler, DD, A&T ... **$399**
TARBELL Floppy Disk Interface Kit **$199**
double density, A&T, List $495 .... **$444**

SHIPPING AND INSURANCE: Add $2.50 for boards, $6 for Selectric converter, $7.50 for Floppy Disk Systems, $15 for Horizons. SHIPPED FREIGHT COLLECT: SuperBrain, Centronics and T.I. printers. Contact us for shipping information on other terminals and printers.

Advertised prices are for prepaid orders. Credit card and C.O.D. 2%higher. Deposit may be required on C.O.D. All prices subject to change and all offers subject to withdrawal without notice.

— WRITE FOR FREE CATALOG —

✓238 **MiniMicroMart, Inc.**

1618 James Street, Syracuse NY 13203 (315) 422-4467 TWX 710-541-0431

kilobaud **Peterborough NH 03458**

MICROCOMPUTING T.M. LIST OF ADVERTISERS

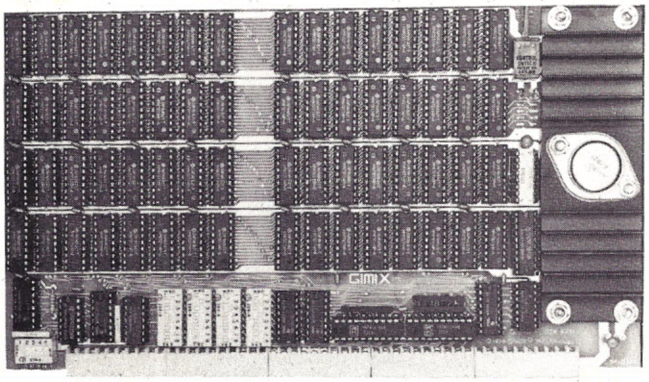
***This advertiser prefers to be contacted directly.**

WANT TO BE KING OF THE HILL?

TREAT YOURSELF ROYALLY WITH GIMIX UNIQUE AND INCOMPARABLE BOARDS AND SYSTEMS... DIP-switch Versatility for use with both SS50 (6800) and SS50C (6809) Systems (SWTP. etc.)

32K STATIC RAM BOARDS

Designed for use with: ★ Existing SS50 Systems
★ SS50C Extended Address Systems



- Assembled
- Burned In
- Tested

16K $328.12

24K $438.14

32K $548.15

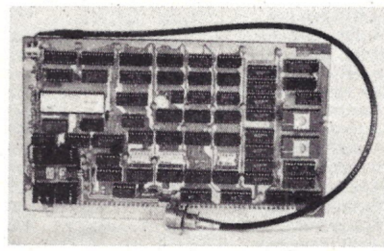
16K and 24K Versions are socketed for 32K and require only additional 2114's for expansion.

FEATURES:

- Decoding for 4 Extended Address Lines (allows memory decoding up to 1 megabyte)
- DIP-switch to set extended addressing or disable it
- 4 separate 8K blocks, addressable to any 8K boundary by DIP-switch
- Each 8K block may be individually disabled
- Write protect either of two 16K sections
- Low power consumption — uses 2114L low power RAMS
- Fully Socketed
- Gold Bus Connectors
- Guaranteed 2MHz operation

FACTORY PRIME STATIC RAMS		
	2114L 450 ns $5.90	200 ns $6.90
	4044 450 ns $5.90	250 ns $6.90

ADD $5.00 HANDLING ON ORDERS UNDER $200.00



THE UNIQUE GIMIX 80 x 24 VIDEO BOARD

★ Upper and Lower Case with Descenders
★ Contiguous 8x10 Character Cells
★ Hardware Scrolling
★ X-Y Addressable Hardware Cursor

It is the ***ONLY* Video Board** that gives you **Software Control** of:

- A programmable RAM Character Generator plus 2 EPROM Character Generators (128 char. ea.).
- Selecting 256 Displayable Characters from 384 available.
- Normal or inverse video, full or reduced intensity, or combinations of these by ***both*** ASCII Code and Bit 8.
- GHOSTability — multiple boards at the same address.

Fully decoded, occupies only 2K of address space
Fully socketed — Gold bus connectors.
Assembled, Burned-in, and Tested at 2MHz.

Deluxe Version **$458.76**
Without RAM Character Generator . **$398.74**
Other Video Boards from **$198.71**

I.O. BOARDS

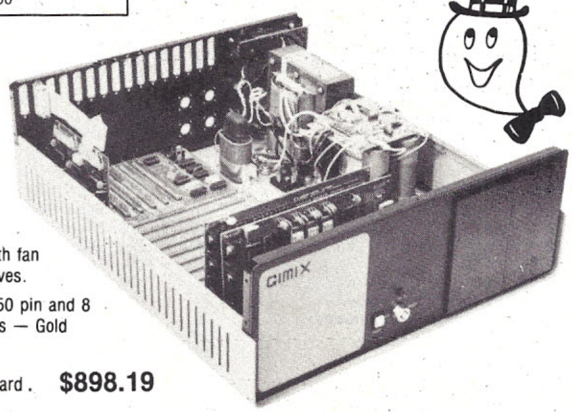
NEW!! 50 PIN 8 PORT RS-232 SERIAL BOARD

★ With SS50C extended addressing (can be disabled)
★ DIP-switch selectable Baud Rates for each Port
★ Occupies 16 bytes of address space
★ Dip switch addressable to any 16 byte boundary
★ Any 2 of IRQ, NMI or FIRQ can be used
★ RS-232 handshaking for each port (CTS, RTS, DCD) **$288.40**
★ With optional on Board Baud Rate Generator **$318.46**

ALSO AVAILABLE

50 Pin 8 Port Parallel **$198.45**
30 Pin 2 Port Parallel . . . . **$ 88.42**
30 Pin 1 Port Serial (RS 232 or 20MA, current loop) . . . . **$ 88.41**
Cable Sets for above . . . . ea. **$ 18.95**

THE CLASSY CHASSIS



- Ferro-resonant Power Supply
- Heavyweight Aluminum Cabinet with fan and provisions for two 5" disk drives.
- 6800/6809 Mother Board, fifteen 50 pin and 8 DIP-switch addressable 30 pin slots — Gold Plated Pins. Fully decoded.
- Baud Rate Generator on Mother Board . **$898.19**

32K SYSTEM *Incomparable Features at a Comparable Price!* **$1,694.59**
Includes: Chassis, 6800 CPU, 32K RAM Board, Choice of I/O Card.

16K Version of above **$1,474.49**

EXPORT NOTES:

For 50Hz 230V C.V. POWER SUPPLY . . . . Add $30.00
80 x 24 VIDEO BOARDS — Specify Format (No Added Charge)
On Orders under $250.00 for a Single Board, or Chips, please Add $30.00 Handling and we will ship Air Mail Prepaid.
On all other orders we will ship via Emery Air Frieght Collect, and we will charge no handling. All orders must be prepaid in U.S. Funds. Please note that foreign checks have been taking about eight weeks for collection, so we would advise wiring money or checks drawn on a bank account in the U.S. Our bank is the Continental Illinois National Bank of Chicago, account #73-32033. Visa or Master Charge also accepted.

GIMIX® and **GHOST®** are Registered Trademarks of GIMIX INC.

Phone, write, or see your dealer for details and prices on our broad range of Boards and Systems for the SS50/SS50C bus and our AC Power Control Products for all computers.

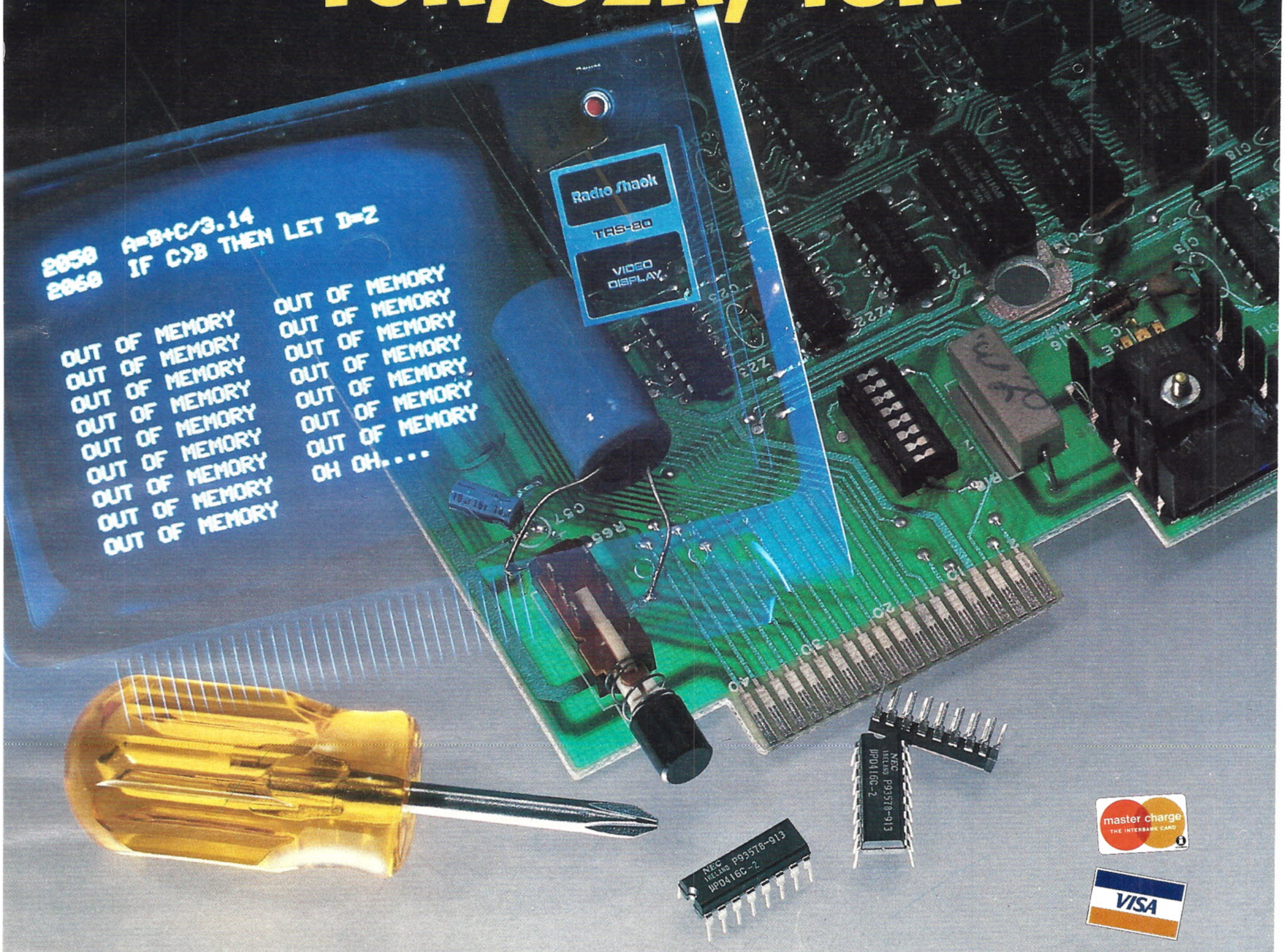
GIMIX INC.

✓22

The Company that delivers
Quality Electronic products since 1975.

1337 WEST 37th PLACE, CHICAGO, IL 60609
(312) 927-5510 • TWX 910-221-4055

Memory Shortage Cures: 16K, 32K, 48K



Dealer Inquiries Invited

The Product. Only high quality, prime, burned-in and tested 4116 16K dynamic RAMs. Don't be caught unaware! All TRS-80 memory expansion kits are not the same. UHF Associates' memory expansion gives you high quality coupled with outstanding performance. And with their fast 200 NS minimum access time (less CPU wait states) UHF's 4116 16K dynamic RAMs provide both storage and speed that won't disappoint you later down the road.

The Price. 16K Memory Expansion Kit for either computer (pre-programmed DIP shunts included) or expansion interface, $95. More? 32K Kit for expansion interface, $180. Most? 48K Kit for computer and expansion interface, $265.

The Promise. "Thou shalt not wait, worry or fret." You'll get immediate post-paid delivery from in-stock inventory. You'll get a full 12 month warranty. That's about four times the warranty others offer. And for installation, you'll get UHF's "goof-proof" instructions. All you'll need is a screwdriver and about 10 minutes.

TRS-80 is a registered trade mark of Tandy Corporation.

- ☐ **16K Kit with shunts** (for computer) $ 95
- ☐ **16K Kit** (for expansion interface) $ 95
- ☐ **32K Kit** (for expansion interface) $180
- ☐ **48K Kit** (for computer and expansion interface) $265

California residents please add appropriate sales tax

Name (print) ______________________

Street ______________________

City ______________ State ________ Zip ________

☐ I've enclosed a check or money order for $ ________ payable to **UHF Associates.**

We honor: ☐ Master Charge ☐ VISA/BankAmericard

Account # ______________________

Expiration Date ______________

Signature ______________________

(required for charge card purchases)

✓2

UHF ASSOCIATES

90 Transport Avenue, #4
Rohnert Park, CA 94928
Call 707/584-7844